{POWER.CODERS}

Objects and arrays

Agenda

Today we will do

- Objects
- Arrays

Recap

What we had so far

What else?

- Variables
- Data types
- Conditions
- Loops
- Functions

OBJECTS

What is an object

An **Object** can contain **many** values and help organize and structure them.

What is an object

An **Object** can contain **many** values and help organize and structure them.

An **Object** is a data type (like numbers or strings), but also a **data structure**.

What is an object

An **Object** can contain **many** values and help organize and structure them.

An **Object** is a data type (like numbers or strings), but also a **data structure**.

An **Object** is a collection of data types. They can even have methods (=functions) in them.

What is an object

An **Object** can contain **many** values and help organize and structure them.

An **Object** is a data type (like numbers or strings), but also a **data structure**.

An **Object** is a collection of data types. They can even have methods (=functions) in them.

Think of a collection as a list of values that are written as **name:value** pairs.

An example

```
var person = {
  name: "John",
  age: 38,
  isMarried: false,
  hello: function() {
    return "Hello " + this.name;
  }
};
```

An example

```
var person = {
  name: "John",
  age: 38,
  isMarried: false,
  hello: function() {
    return "Hello " + this.name;
  }
};
```

- Each line, separated by a comma, is called a **property**.
- The names of the variables on the left are called **property names**.
- The values on the right side are called **property values**.

An example

```
var person = {
  name: "John",
  age: 38,
  isMarried: false,
  hello: function() {
    return "Hello " + this.name;
  }
};
```

- Each line, separated by a comma, is called a **property**.
- The names of the variables on the left are called **property names**.
- The values on the right side are called **property values**.

Javascript objects are containers for named values.

Interact with Objects

There are 2 ways to access **object properties**:

```
person.age;
person["age"];
```

There is only 1 way to access a **object method**:

```
person.hello();
//if you type person.hello without () you get back the defintion back
```

Another example

```
document.write(person.name.length);
```

ANOTHER EXAMPLE

```
document.write(person.name.length);
```

The **write()** function is actually a method of the **document** object.

Another example

```
document.write(person.name.length);
```

The **write()** function is actually a method of the **document** object.

The built-in **length** property is used to count the number of characters.

null

You can empty an object by setting it to null.

```
person = null;
person.name = "Susanne"; // not possible
```

REMEMBER

```
var person = {
  name: "John",
  age: 38,
  isMarried: false,
  hello: function() {
    return "Hello " + this.name;
  }
};
```

REMEMBER

```
var person = {
  name: "John",
  age: 38,
  isMarried: false,
  hello: function() {
    return "Hello " + this.name;
  }
};
```

Initializing the object this way, created **one single object**.
But David is a person, too.

REMEMBER

```
var person = {
  name: "John",
  age: 38,
  isMarried: false,
  hello: function() {
    return "Hello " + this.name;
  }
};
```

Initializing the object this way, created **one single object**. But David is a person, too.

We need a more general object type that can be used to create a number of objects of the same type.

Object constructor

```
function person(name, age, married) {
  this.name = name;
  this.age = age;
  this.isMarried = married;
  this.hello = function() {
    return "Hello " + this.name;
  }
};
```

An object constructor is a function that performs the task of defining an object.

The `this` keyword refers to the **current object**.

You also need `this` to access the variables of your own object, e.g. **inside a method**.

Creating instances of an object

Once you have a object constructor, use the `new` keyword to create a new object of the same type called **instance**.

```
let susanne = new person("Susanne", 38, false);
let max = new person("Max", 45, true);

document.write(susanne.age); // Output: 38
document.write(max.hello()); // Output: Hello Max

susanne.age = 45; // Possible to change a property value
document.write(susanne.age); // Output: 45

susanne.gender = "female"; // Possible to add new property
document.write(susanne.gender); // Output: female
delete susanne.gender; // Possible to delete property
```

What keyword is used for creating an instance of an object?

`Object` object

`Object.keys()` lists all property names of an object in an array.

`Object.values()` returns all property values of an object in an array.

```
Object.keys(susanne);
// ["name", "age", "isMarried", "hello"]
```

JAVASCRIPT ARRAYS

What is an array

Arrays store multiple values in a single variable.

```
var topics = ["HTML","CSS","JS"];
```

What is an array

Arrays store multiple values in a single variable.

```
var topics = ["HTML","CSS","JS"];
```

- An array is a **special type of object**.

What is an array

Arrays store multiple values in a single variable.

```
var topics = ["HTML","CSS","JS"];
```

- An array is a **special type of object**.
- An array is a collection of often similar data.

What is an array

Arrays store multiple values in a single variable.

```
var topics = ["HTML","CSS","JS"];
```

- An array is a **special type of object**.
- An array is a collection of often similar data.
- An array can hold any data in JS: objects, numbers, strings ...

What is an array

Arrays store multiple values in a single variable.

```
var topics = ["HTML","CSS","JS"];
```

- An array is a **special type of object**.
- An array is a collection of often similar data.
- An array can hold any data in JS: objects, numbers, strings ...
- A **multidimensional array** contains arrays.

Accessing an array

You access an array element by referring to the **index number** written in **square brackets**.

```
var firstTopic = topics[0];

topics[2] = "jQuery";
// Possible to overwrite a value in an array
```

Accessing an array

You access an array element by referring to the **index number** written in **square brackets**.

```
var firstTopic = topics[0];

topics[2] = "jQuery";
// Possible to overwrite a value in an array
```

- Array indexes start with **0**

Accessing an array

You access an array element by referring to the **index number** written in **square brackets**.

```
var firstTopic = topics[0];

topics[2] = "jQuery";
// Possible to overwrite a value in an array
```

- Array indexes start with **0**
- Referring to an index outside of the array, returns **undefined**

Accessing an array

You access an array element by referring to the **index number** written in **square brackets**.

```
var firstTopic = topics[0];

topics[2] = "jQuery";
// Possible to overwrite a value in an array
```

- Array indexes start with **0**
- Referring to an index outside of the array, returns **undefined**
- An array uses **numbers** to access its elements, an object uses **names**

Creating arrays

There are several ways how to declare an array

```
var topics = new Array(3);
topics[0] = "HTML";
topics[1] = "CSS";
topics[2] = "JS";
```

```
var topics = new Array(); // more dynamic without argument
topics[0] = "HTML";
topics[1] = "CSS";
topics[2] = "JS";
topics[3] = "PHP";
```

```
var topics = ["HTML","CSS","JS"] // recommended way to declare arrays
```

`length`

- Remember: `length` is a built-in JS property of any object.
- `length` returns the number of items inside an array.
- The value of `length` is always one more than the highest index.
- If the array is empty, the `length` property returns **0**.

Combining arrays

The `concat` method takes two arrays and combines them in **one new array**.

```
var t1 = ["HTML","CSS"];
var t2 = ["JS","PHP"]
var topics = t1.concat(t2);
```

More array methods

Next to `concat` these are some of the most important methods you can use:

More array methods

Next to `concat` these are some of the most important methods you can use:

- `topics.toString()` This will return all the elements as a string seperated by a comma.

More array methods

Next to `concat` these are some of the most important methods you can use:

- `topics.toString()` This will return all the elements as a string seperated by a comma.
- `topics.push("MySQL")` This will add "MySQL" at the end of the array.

More array methods

Next to `concat` these are some of the most important methods you can use:

- `topics.toString()` This will return all the elements as a string seperated by a comma.
- `topics.push("MySQL")` This will add "MySQL" at the end of the array.
- `topics.pop()` This will return the last element of the array and will remove it.

More array methods

More array methods

- `topics.shift()` This will return the first element of the array and will remove it.

More array methods

- `topics.shift()` This will return the first element of the array and will remove it.
- `topics.sort()` This will sort the array alphabetically.

More array methods

- `topics.shift()` This will return the first element of the array and will remove it.
- `topics.sort()` This will sort the array alphabetically.

and many more

Multidimensional Array

In Javascript a multidimensional array is an array where each element is also an array.

```
let timeSpent = [
    ['Work', 9],
    ['Eat', 2],
    ['Commute', 1],
    ['Watch TV', 2],
    ['Sleep', 7]
];

document.write(timeSpent[0][1]);
```

Multidimensional Array

In Javascript a multidimensional array is an array where each element is also an array.

```
let timeSpent = [
    ['Work', 9],
    ['Eat', 2],
    ['Commute', 1],
    ['Watch TV', 2],
    ['Sleep', 7]
];

document.write(timeSpent[0][1]);
```

To access an element of the multidimensional array, you first use square brackets to access an element of the outer array which returns an inner array; and then use another square bracket to access the element of the inner array.

Built-in objects

- `Math` to perform mathematical tasks
- `Date` to work with dates

Math

```
let pi = Math.PI;
document.write(Math.floor(pi));
document.write(Math.round(pi));
document.write(Math.ceil(pi));

let randomNumber = Math.ceil(Math.random() * 10);
document.write(randomNumber);
```

Date

```
function printTime(){
  let currentDate = new Date();
  let hours = currentDate.getHours();
  let mins = currentDate.getMinutes();
  let secs = currentDate.getSeconds();

  document.write(hours + ":" + mins + ":" + secs +"\n");
}

setInterval(printTime, 1000); // prints current time each second
```

Online ressources

Check what we learned so far on w3schools.com and mdn.com, e.g.

- functions
- objects
- arrays
- Data structures

EXERCISES

Exercises

- Try them yourself first (20 min each)
- Then discuss your solutions with your buddy (10 min each)

Exercise: Mad Lib

Do your Mad Lib exercise again:

Create a simple mad-lib program that prompts for a noun, a verb, an adverb, and an adjective and injects those into a story that you create.

Write a function and store the variables in an array - or better an object?

Exercise: Fortune Teller

Do the Fortune Teller again but this time with a object.

Write a function named tellFortune that:

- takes 4 arguments: number of children, partner's name, geographic location, job title.
- outputs your fortune to the screen like so: "You will be a X in Y, and married to Z with N kids."
- Call that function 3 times with 3 different values for the arguments.

Exercise: Your top choices

- Create an array to hold your top choices (colors, presidents, whatever).
- For each choice, log to the screen a string like: "My #1 choice is blue."
- Change your top choices into objects, e.g. presidents with name, year of presidency, etc as properties.

Work on your project