

·⚙· thinkdev #3

Collections

Let's revisit the FutureLearn courses example:

Explore featured courses



The University of Glasgow ☆

The Museum as a Site and Source for Learning

★★★★☆ 4.6 (75 reviews)

[Find out more](#)



Sentinel|9 & FutureLearn ☆

NEW

Fundamentals of Business Strategy

[Find out more](#)



University of Groningen,
University of Cambridge &
University Medical Center
Groningen (UMCG) ☆

Young People and Mental Health

★★★★☆ 4.7 (649 reviews)

[Find out more](#)



Part of an ExpertTrack

Coventry University ☆

International Logistics: A Beginner's Guide to Logistics Management

[Find out more](#)

How do we represent it in code?

```
let courseTitle = 'The Museum as a Site and ...'  
let courseRating = 4.6  
let courseReviewsCount = 75  
let courseIsNew = false  
let courseIsPartOfAnExpertTrack = false
```

```
let courseTitle = 'The Museum as a Site and ...'  
let courseRating = 4.6  
let courseReviewsCount = 75  
let courseIsNew = false  
let courseIsPartOfAnExpertTrack = false
```

```
let course2Title = 'Fundamentals of Business ...'  
let course2Rating = 0  
let course2ReviewsCount = 0  
let course2IsNew = true  
let course2IsPartOfAnExpertTrack = false
```

```
let course3Title = 'Young People and Mental Health'
```

```
let course3Rating = 4.7
```

```
let course3ReviewsCount = 649
```

```
let course3IsNew = false
```

```
let course3IsPartOfAnExpertTrack = false
```

```
let course4Title = 'International Logistics: A ...'
```

```
let course4Rating = 0
```

```
let course4ReviewsCount = 0
```

```
let course4IsNew = false
```

```
let course4IsPartOfAnExpertTrack = false
```

**Things are getting unwieldy already; we have
so many related variables that are not tied
together 😞.**

**We need a better way to represent “entities”
that have different “attributes”.**

Objects

Let's break it down

Start with curly brackets:

```
const course = {}
```

Add a property (a key: value pair):

```
const course = {  
  title: 'The Museum as a Site and ...'  
}
```

Add more properties; separate by commas (last comma is optional):

```
const course = {  
  title: 'The Museum as a Site and ...',  
  rating: 4.6,  
  reviewsCount: 75,  
  isNew: false,  
  isPartOfAnExpertTrack: false,  
}
```

How to name properties

Use strings:

```
const obj = {  
  "prop": "...",  
  "another prop": "...",  
  "&@+/" : "...",  
  "0": "...",  
}
```


The quotes are optional if the name is a valid variable name (i.e. an *identifier*) or a number:

```
const obj = {  
  prop: "...",  
  "another prop": "...",  
  "&@+/" : "...",  
  0: "...",  
}
```

The JavaScript way is also camelCase:

```
const obj = {  
  prop: "...",  
  anotherProp: "...",  
}
```

Property names must be unique:

```
const obj = {  
  prop: 1,  
  prop: 2, // this overrides the previous one  
}
```

// same as

```
const obj = {  
  prop: 2  
}
```

Property names must be unique:

```
const obj = {  
  prop: 1,  
  prop: 2, // this overrides the previous one  
}
```

// same as

```
const obj = {  
  prop: 2  
}
```

Using objects

Access a property

Get the value of a property using the dot notation:

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  'reviews count': 0,  
  isNew: true,  
}  
  
console.log(course.title) // The Fundamentals of ...
```

Set the value of a property too with the dot notation:

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  'reviews count': 0,  
  isNew: true,  
}  
  
course.isNew = false  
  
console.log(course.isNew) // false
```

Be careful with property names that aren't identifiers:

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  'reviews count': 0,  
  isNew: true,  
}  
  
course.'reviews count'++ // Error
```


Use the bracket notation instead for such properties:

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  'reviews count': 0,  
  isNew: true,  
}  
  
course['reviews count']++  
  
console.log(course['reviews count']) // 1
```

You can add a property:

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  'reviews count': 0,  
  isNew: true,  
}  
  
course.rating = 0  
  
console.log(course.rating) // 0
```

And delete a property too:

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  'reviews count': 0,  
  isNew: true,  
}
```

```
delete course['reviews count']
```

```
console.log(course['reviews count']) // undefined
```

Objects are *mutable*

In contrast, numbers, strings, and booleans are *immutable*:

```
const str = "Strings are immutable"

// Try to change 'Strings' to 'Springs'
str[1] = 'p' // No error, but it doesn't work

console.log(str) // Strings are immutable
```

Does an object have a property?

Use the `in` operator to check if an object has a property:

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  'reviews count': 0,  
  isNew: true,  
}
```

```
console.log("title" in course) // true  
console.log("rating" in course) // false
```

Pack variables into an object

It's common to have variables that you want to pack into an object:

```
const title = 'The Fundamentals of Business Strategy'  
const reviewsCount = 0  
const isNew = true
```

You can set the object properties manually:

```
const title = 'The Fundamentals of Business Strategy'  
const reviewsCount = 0  
const isNew = true  
  
const course = {  
  title: title,  
  reviewsCount: reviewsCount,  
  isNew: isNew,  
}
```


Or use the shorthand form:

```
const title = 'The Fundamentals of Business Strategy'  
const reviewsCount = 0  
const isNew = true  
  
const course = {  
  title,  
  reviewsCount,  
  isNew,  
}
```

How about unpacking?

It may be tedious to type the `course` . prefix sometimes

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  reviewsCount: 0,  
  isNew: true,  
}
```

You can unpack the properties you need into variables:

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  reviewsCount: 0,  
  isNew: true,  
}
```

```
const title = course.title  
const isNew = course.isNew
```

There's also a shorter way; it's called *destructuring*:

```
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  reviewsCount: 0,  
  isNew: true,  
}  
  
const { title, isNew } = course
```

Copy an object into a new one

You may want to copy the properties of one object into a new one:

```
const ratingInfo = {  
  rating: 0,  
  reviewsCount: 0,  
}  
  
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  isNew: true,  
  // You want rating and reviewsCount here.  
}
```

Here's one way to do it:

```
const ratingInfo = {  
  rating: 0,  
  reviewsCount: 0,  
}  
  
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  isNew: true,  
  rating: ratingInfo.rating,  
  reviewsCount: ratingInfo.reviewsCount,  
}
```

Another way is to *spread* the object:

```
const ratingInfo = {  
  rating: 0,  
  reviewsCount: 0,  
}  
  
const course = {  
  title: 'The Fundamentals of Business Strategy',  
  isNew: true,  
  ...ratingInfo,  
}
```

**Let's update the original example
to use objects**

**Now we have to deal with many
separate but similar entities.**

We need a “list” of some sort.

Arrays

The elements can be of different types:

```
const arr = ["hi", 12.34, true]
```

Access an array element

Arrays are ordered and can be indexed, like strings:

```
//           0           1           2
const people = ["Amal", "Isa", "Khadija"]

console.log(people[0]) // "Amal"

// Replace "Isa" with "Elleman"
people[1] = "Elleman"
```

Access an array element

Arrays are ordered and can be indexed, like strings:

```
//           0           1           2
const people = ["Amal", "Isa", "Khadija"]

console.log(people[0]) // "Amal"

// Replace "Isa" with "Elleman"
people[1] = "Elleman"
```

How long is this array?

Use the `length` property to get the length of an array.

```
const people = ["Amal", "Isa", "Khadija"]  
  
console.log(people.length) // 3
```

Push to an array

Use the push method to add an item to the end of an array:

```
const people = ["Amal", "Isa", "Khadija"]  
  
people.push("Mubaraq")  
  
console.log(people)  
// ["Amal", "Isa", "Khadija", "Mubaraq"]
```

Pop from an array

Use the pop method to remove the last element of an array:

```
const people = ["Amal", "Isa", "Khadija"]  
  
people.pop()  
  
console.log(people)  
// ["Amal", "Isa"]
```

Does an array have an element?

The `includes` method tells if an array contains a certain element:

```
const people = ["Amal", "Isa", "Khadija"]  
  
people.includes("Isa") // true  
people.includes("Mubaraq") // false
```

Get a portion of an array

Use the `slice` method:

```
//           0       1       2
const people = ["Amal", "Isa", "Khadija"]

people.slice(0, 2) // ["Amal", "Isa"]
people.slice(1)   // ["Isa", "Khadija"]
```


Get a portion of an array

Use the `slice` method:

```
//           0       1       2
const people = ["Amal", "Isa", "Khadija"]

people.slice(0, 2) // ["Amal", "Isa"]
people.slice(1)   // ["Isa", "Khadija"]
```

Spread an array into another

```
const names = ["Aisha", "Mubaraq"]  
const people = ["Amal", "Isa", "Khadija", ...names]  
  
console.log(people)  
// ["Amal", "Isa", "Khadija", "Aisha", "Mubaraq"]
```

Arrays are also mutable ...

... because they are objects.

```
const people = ["Amal", "Isa", "Mubaraq"]
```

```
console.log(typeof people)
```

```
// object 🤪
```

Finally ...


```
isNew: true,  
isPartOfAnExpertTrack: false,  
},  
{  
  title: 'Young People and Mental Health',  
  rating: 4.7,  
  reviewsCount: 649,  
  isNew: false,  
  isPartOfAnExpertTrack: false,  
},  
{  
  title: 'International Logistics: A  
  '
```

