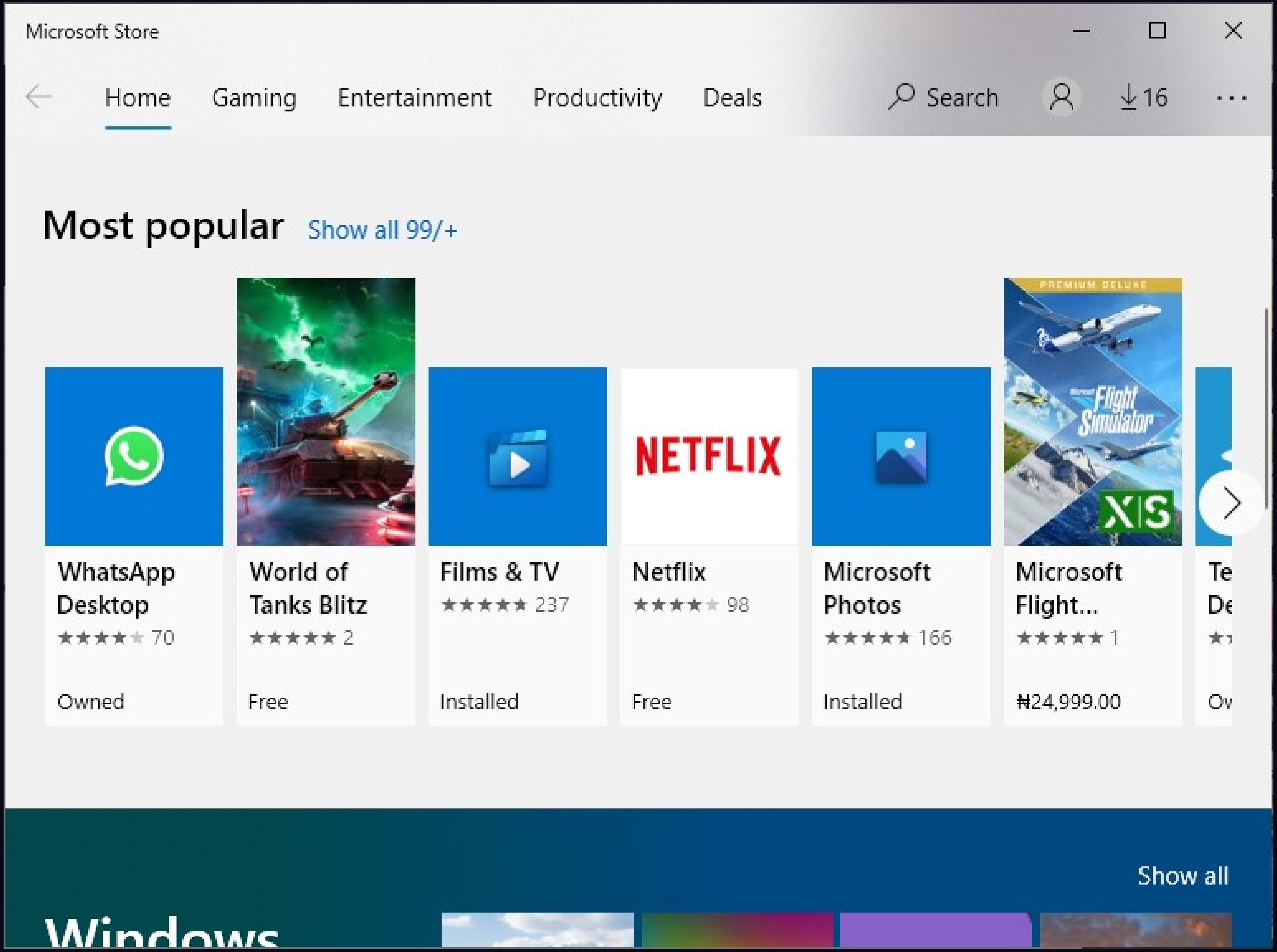
thinkdev #3

Collections

The store example



Most popular Show all 99/+

WhatsApp Desktop	★★★★★ 70	Owned
World of Tanks Blitz	★★★★★ 2	Free
Films & TV	★★★★★ 237	Installed
Netflix	★★★★★ 98	Free
Microsoft Photos	★★★★★ 166	Installed
Microsoft Flight...	★★★★★ 1	₦24,999.00

Windows

How to represent in code?

```
let appName = 'WhatsApp Desktop'
let appRating = 4
let appReviewsCount = 70
let appPrice = 0
let appStatus = 'Owned' // or 'Installed' or 'Not Owned'
```

How to represent in code?

```
let appName = 'WhatsApp Desktop'
let appRating = 4
let appReviewsCount = 70
let appPrice = 0
let appStatus = 'Owned' // or 'Installed' or 'Not Owned'

let app2Name = 'World of Tank Blitz'
let app2Rating = 5
let app2ReviewsCount = 2
let app2Price = 0
let app2Status = 'Not Owned'
```

How to represent in code?

Problem: so many variables that aren't tied together

We need a way to represent entities with different parts

Objects

```
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  price: 0,
  status: 'Owned',
}
```

Let's break it down

Let's break it down

Start with curly brackets

```
const app = {}
```

Let's break it down

Add a property (`key: value`)

```
const app = {
  name: 'WhatsApp Desktop'
}
```

Let's break it down

Add more properties, separated by commas (last comma is optional)

```
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  price: 0,
  status: 'Owned',
}
```

How to name a property?

How to name a property?

Use a string

```
const obj = {
  "prop": "...",
  "another prop": "...",
  "$#@!": "...",
  "0": "...",
}
```

How to name a property?

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

```
const obj = {
  prop: "...",
  "another prop": "...",
  "$#@!": "...",
  0: "...",
}
```

How to name a property?

The JavaScript way is also `camelCase`

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

How to name a property?

Property names must be unique.

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

// same as
const obj = {
  prop: 2
}
```

Using objects

Get/set a property

Dot notation

```
const app = {
  name: 'WhatsApp Desktop',
  status: 'Owned',
  123: null,
}

// Get a prop
console.log(app.name) // 'WhatsApp Desktop'

// Set (i.e. change) a prop
app.status = 'Installed'
```

Get/set a property

Bracket notation. For property names that aren't valid identifiers

```
const app = {
  name: 'WhatsApp Desktop',
  status: 'Owned',
  123: null,
}

// Get a prop
console.log(app['name']) // 'WhatsApp Desktop'

// Set (i.e. change) a prop
app['status'] = 'Installed'
```

Add/remove a property

```
const app = {
  name: 'WhatsApp Desktop',
  status: 'Owned',
  123: null,
}

// Add a prop
app.rating = 4

// Remove a prop
delete app[123]
```

Objects are ***mutable***

- You can't change a number, string, or boolean intrinsically, because they are *immutable*.
- But you can change "parts" of an object!

Objects are *mutable*

```
const str = "Strings are immutable"

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

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

Does an object have a property?

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

```
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  price: 0,
  status: 'Owned',
}

console.log("rating" in app) //true
console.log("downloadsCount" in app) // false
```

Pack variables into an object

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

```
const name = 'WhatsApp Desktop'
const rating = 4
const reviewsCount = 70
const price = 0
const status = 'Owned'
```

Pack variables into an object

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

```
const name = 'WhatsApp Desktop'
const rating = 4
const reviewsCount = 70
const price = 0
const status = 'Owned'

const app = {
  name: name,
  rating: rating,
  reviewsCount: reviewsCount,
  price: price,
```

Pack variables into an object

There's a shorter way

```
const name = 'WhatsApp Desktop'
const rating = 4
const reviewsCount = 70
const price = 0
const status = 'Owned'

const app = {
  name,
  rating,
  reviewsCount,
  price,
```

How about unpacking?

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

```
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  price: 0,
  status: 'Owned',
}
```

How about unpacking?

We can unpack the properties we need into variables

```
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  price: 0,
  status: 'Owned',
}

const name = app.name
const rating = app.rating
// ...
```

How about unpacking?

There's also a shorter way! It's called *destructuring*.

```
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  price: 0,
  status: 'Owned',
}

const { name, rating } = app
```

Reuse an object to create another

Sometimes you want to create a new object with the properties of an existing object.

```
const purchaseInfo = {
  price: 0,
  status: 'Owned',
}
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  // We want price and status here
}
```

Reuse an object to create another

One way to do it:

```
const purchaseInfo = {
  price: 0,
  status: 'Owned',
}
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  price: purchaseInfo.price,
  status: purchaseInfo.status,
}
```

Reuse an object to create another

Another way: *spread* the object with . . .

```
const purchaseInfo = {
  price: 0,
  status: 'Owned',
}
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  ...purchaseInfo,
}
```

We've solved the issue of tying different parts to form entities ...

… but how do we deal with so many similar entities?

```
const app = {
  name: 'WhatsApp Desktop',
  rating: 4,
  reviewsCount: 70,
  price: 0,
  status: 'Owned',
}

const app2 = {
  name: 'World of Tank Blitz',
  rating: 5,
```

We need a "list" of some sort ...

Arrays

Arrays

An *array* is a sequence of elements.

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

Arrays

Arrays are ordered and can be indexed, like strings

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

people[0] // "Amal"

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

Arrays

The elements can be of different types.

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

How long is this array?

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

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

people.length // 3
```

Pop and push

Use the pop method to remove from the end,
and push to add to the end.

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

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

people.push("Ibrahim")
console.log(people) // ["Amal", "Isa", "Ibrahim"]
```

Shift, unshift

Like pop and push, but at the front of an array.

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

const removed = people.shift()
console.log(removed) // "Amal"
console.log(people) // ["Isa", "Mubaraq"]

people.unshift("Aisha")
console.log(people) // ["Aisha", "Isa", "Mubaraq"]
```

Does an array have an element?

Use the `includes` method to check if an array contains a certain element.

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

people.includes("Isa") // true
people.includes("Ibrahim") // false
```

Get a portion of an array

Use the `slice` method.

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

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

// Create a copy of the array
people.slice() // ["Amal", "Isa", "Mubaraq"]
```

Spread an array into another

```
const names = ["Aisha", "Ibrahim"]
const people = ["Amal", "Isa", "Mubaraq", ...names]

console.log(people)
// ["Amal", "Isa", "Mubaraq", "Aisha", "Ibrahim"]
```

Unpack an array

Unpack array elements into variables.

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

const first = people[0] // "Amal"
const second = people[1] // "Isa"
```

Unpack an array

Another way is to *destructure*

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

const [first, second] = people
// first is "Amal", second is "Isa"

// You can skip items too
const [, , third] = people
// third is "Mubaraq"
```

Arrays are also mutable ...

… because they are objects

... because they are objects

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

typeof people
// "object" 😕
```

How do we check if something is an array?

How do we check if something is an array?

Use the `Array.isArray` method.

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

Array.isArray(people)
// true
```

Finally ...

```
const apps = [
  {
    name: 'WhatsApp Desktop',
    rating: 4,
    reviewsCount: 70,
    price: 0,
    status: 'Owned',
  },
  {
    name: 'World of Tank Blitz',
    rating: 5,
```

Exercises

Check the website

Questions?