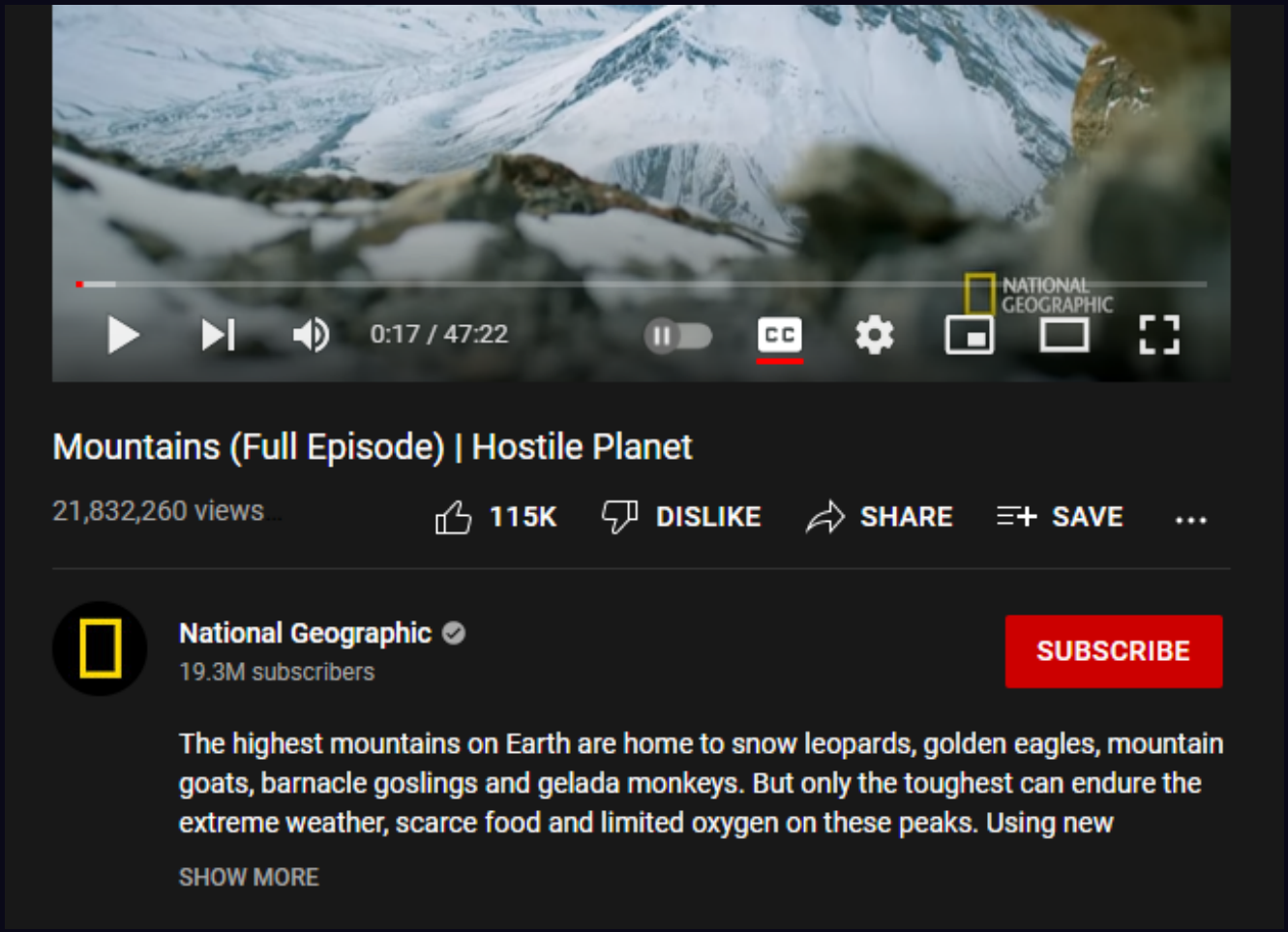
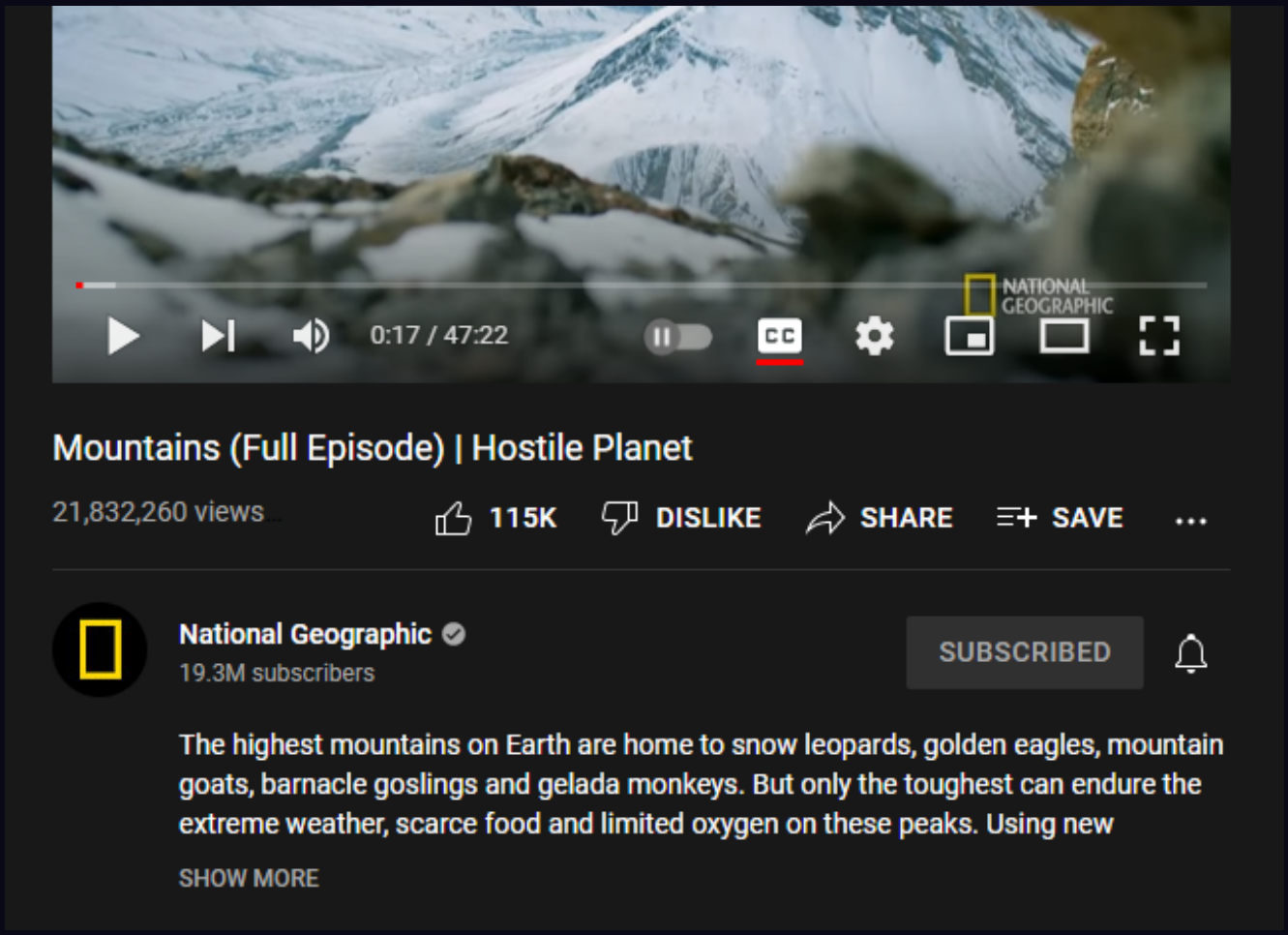
thinkdev #4

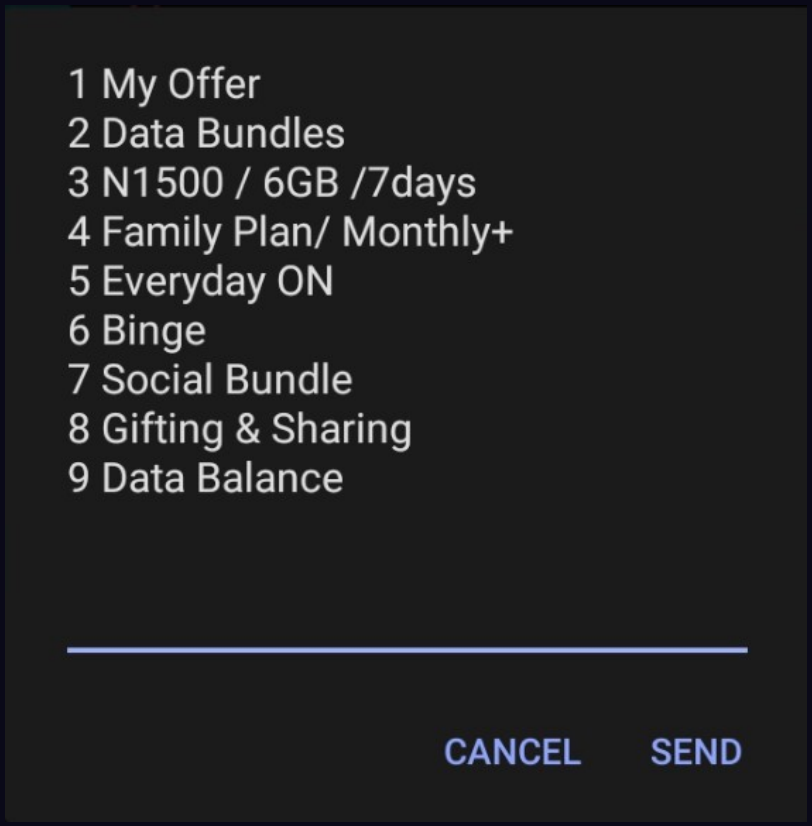
Making decisions



Screenshot of a YouTube video, showing the channel subscribe button



Screenshot of a YouTube video, showing the channel subscribed and notification buttons



A USSD application menu showing the different options a user can choose from.

First, let's learn how to do some basic comparison

Relational operators

```
3 > 2 // true
8 < 5 // false
9 >= 13 // false
6 <= 6 // true
```

Equality

Check if two values are equal with the strict equality operator ===

```
2 === 5 - 3 // true
'hello' === 'hi' // false
true === false // false
'10' === 10 // false
89.0 === 89 // true
```

Inequality

Check if two values are *not* equal with the strict inequality operator `!==`

```
2 !== 5 - 3 // false
'hello' !== 'hi' // true
true !== false // true
'10' !== 10 // true
89.0 !== 89 // false
```

Equal objects?

No two objects have the same value, even if they look alike.

```
const obj1 = { prop: "value" }
const obj2 = { prop: "value" }
console.log(obj1 === obj2) // false 😕
```

Equal objects?

But, same object, same value.

```
const obj1 = { prop: "value" }
const obj2 = obj1
console.log(obj1 === obj2) // true
```

Truthy and falsy values

Truthy and falsy values

We can classify values based on their "truth".

Truthy and falsy values

A value that converts to the boolean `true` is *truthy*.

```
Boolean(10) // true
Boolean("think" + "dev") // true
Boolean({ x: 5 }) // true
```

Truthy and falsy values

A value that converts to the boolean `false` is *falsy*.

```
Boolean(0) // false
Boolean(null) // false
Boolean("") // false
```

Truthy and falsy values

All values are truthy except a few which are falsy:

Truthy and falsy values

All values are truthy except a few which are falsy:

- `false`

Truthy and falsy values

All values are truthy except a few which are falsy:

- `false`
- `0`

Truthy and falsy values

All values are truthy except a few which are falsy:

- `false`
- `0`
- `" "` (empty string)

Truthy and falsy values

All values are truthy except a few which are falsy:

- `false`
- `0`
- `""` (empty string)
- `null`

Truthy and falsy values

All values are truthy except a few which are falsy:

- `false`
- `0`
- `""` (empty string)
- `null`
- `undefined`

Let's get to making decisions now

The `if` statement

The `if` statement

```
if (expression) {
  statement1
  statement2
  ...
}
```

The `if` statement

```
if (expression) {
  statement1
  statement2
  ...
}
```

If `expression` is truthy, execute the statements in the curly brackets.

Let's consider the YouTube example

We could represent a user like so:

```
const user = {
  name: "Mubaraq Wahab",
  subscriptions: ["National Geographic", "Elleman10"],
  // ...
}
```

And determine if they're subscribed to a certain channel.

```
const user = {
  name: "Mubaraq Wahab",
  subscriptions: ["National Geographic", "Elleman10"],
  // ...
}

user.subscriptions.includes('National Geographic')
```

Then we can act accordingly:

```
const user = {
  name: "Mubaraq Wahab",
  subscriptions: ["National Geographic", "Elleman10"],
  // ...
}

if (user.subscriptions.includes('National Geographic'))
  console.log('You are subscribed')
}
```

Let's add some logs around the `if` statement for clarity.

```
const user = {
  name: "Mubaraq Wahab",
  subscriptions: ["National Geographic", "Elleman10"],
  // ...
}

console.log('Before decision')
if (user.subscriptions.includes('National Geographic'))
  console.log('You are subscribed')
}
console.log('After decision')
```

Output if the user is subscribed to National Geographic:

```
Before decision
You are subscribed
After decision
```

Otherwise, ...

```
const user = {
  name: "Mubaraq Wahab",
  subscriptions: ["Elleman10"],
  // ...
}

console.log('Before decision')
if (user.subscriptions.includes('National Geographic'))
  console.log('You are subscribed')
}
console.log('After decision')
```

... the output is just this

```
Before decision
After decision
```

... the output is just this

```
Before decision
After decision
```

How do we print a different message?

else

else

```
console.log('Before decision')
if (user.subscriptions.includes('National Geographic'))
  console.log('You are subscribed')
} else {
  console.log('You are not subscribed')
}
console.log('After decision')
```

else

The result?

```
Before decision
You are not subscribed
After decision
```

Let's consider another example

```
const n = 3;

if (n > 0) {
  console.log(n, 'is positive')
} else {
  console.log(n, 'is negative')
}
```

Let's consider another example

```
const n = 3;

if (n > 0) {
  console.log(n, 'is positive')
} else {
  // Wrong: n could be zero.
  console.log(n, 'is negative')
}
```

else if

```
const n = 3;

if (n > 0) {
  console.log(n, 'is positive')
} else if (n < 0) {
  console.log(n, 'is negative')
}
```

else if

```
const n = 3;

if (n > 0) {
  console.log(n, 'is positive')
} else if (n < 0) {
  console.log(n, 'is negative')
} else {
  console.log(n, 'is zero')
}
```

else if

We can have many `else if`s too.

```
const n = 3;

if (n > 0) {
  console.log(n, 'is positive')
} else if (n < 0) {
  console.log(n, 'is negative')
} else if (n === 0) {
  console.log(n, 'is zero')
} else {
  // do something else
}
```

We can now make decisions based on simple conditions

But what if we have complex conditions?

But what if we have complex conditions?

E.g., accept an uploaded file if it's an image
and it's not larger than 2MB

We'll use logical operators

We'll use logical operators

NOT

We'll use logical operators

NOT, AND

We'll use logical operators

NOT, AND, and OR.

NOT

`!expr`

- The result is `false` if `expr` is truthy.
- The result is `true` if `expr` is falsy.

NOT

```
const arr = []

console.log(arr.length) // 0, which is falsy

if (!arr.length) {
  console.log("The array is empty")
}
```

NOT

You can use it to convert a value to a boolean:

```
null            // falsy value
!null           // true
!!null          // false

// Same as:
Boolean(null) // false
```

AND

`expr1 && expr2`

Both expressions must be truthy for the result to be truthy.

AND

```
if (user && user.role === "ADMIN") {
  // Show something only admins should see
}
```

OR

```
expr1 || expr2
```

At least one of the expressions must be truthy for the result to be truthy.

OR

```
if (filename.endsWith(".docx") || filename.endsWith(".do
  console.log(filename, "is a Word document")
}
```

OR

It's fine to break long lines

```
if (
  filename.endsWith(".docx") ||
  filename.endsWith(".doc")
) {
  console.log(filename, "is a Word document")
}
```

One final thing ...

Remember that `if` is a *statement*

Remember that `if` is a *statement*

So the following is invalid

```
const n = 8

// Error
const remark = if (n % 2 === 0) {
  "It is even"
} else {
  "It is odd"
}
```

Remember that `if` is a *statement*

And we would have to do this instead

```
const n = 8

let remark

if (n % 2 === 0) {
  remark = "It is even"
} else {
  remark = "It is odd"
}
```

But JS has an "`if` expression" too ...

The conditional operator

The conditional operator

```
expr1 ? expr2 : expr3
```

If `expr1` is truthy, the result is `expr2`.
Otherwise, the result is `expr3`.

The conditional operator

```
const n = 8;
const remark = n % 2 === 0 ? "It is even" : "It is odd"
console.log(remark) // "It is even"
```

The conditional operator

- It's called a *ternary* operator, because it operates on three expressions.

The conditional operator

- It's called a *ternary* operator, because it operates on three expressions.
- Similarly, the NOT operator is a *unary* operator because it operates on a single expression, while the AND and OR operators are *binary* because they operate on two expressions.

Questions?