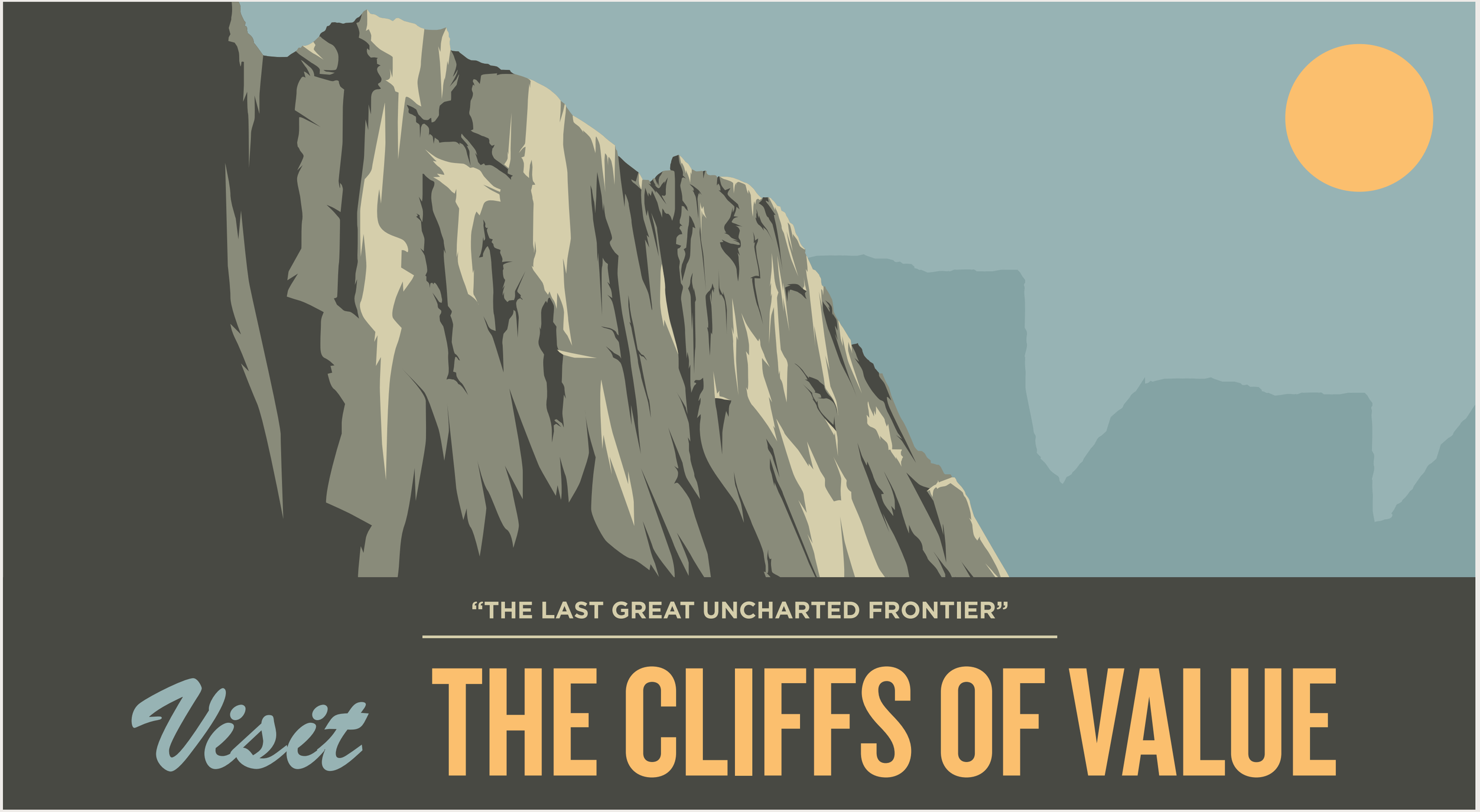

"THE LAST GREAT UNCHARTED FRONTIER"
Visit
THE CLIFFS OF VALUE

LEVEL 1

THE CLIFFS OF VALUE

THE PROMPT & SOME BASIC NUMBERS

`>` The JavaScript Prompt, aka “the Console”

→ What gets returned from the code

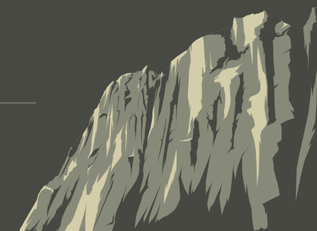
JavaScript automatically recognizes numbers

```
> 24
```

→ `24`

```
> 3.14
```

→ `3.14`



OPERATORS

Common Operators used in JavaScript Syntax:

addition

```
> 6 + 4
→ 10
```

subtraction

```
> 9 - 5
→ 4
```

multiplication

```
> 3 * 4
→ 12
```

division

```
> 12 / 4
→ 3
```

modulus

```
> 43 % 10
→ 3
```

Modulus returns the remainder after division.

ORDER OF OPERATIONS: PEMDAS

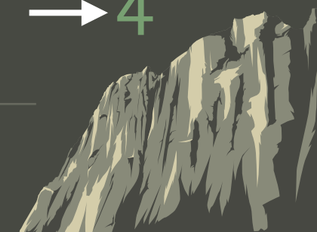
Grouping Expressions in JavaScript

```
> (5 + 7) * 3
    12    * 3
          → 36
```

```
> (3 * 4) + 3 - 12 / 2
    12    + 3 - 12 / 2
    12    + 3 -    6
                → 9
```

```
> (-5 * 6) - 7 * -2
   -30     - 7 * -2
   -30     -   -14
               → -16
```

```
> 4 + (8 % (3 + 1))
  4 + (8 %    4   )
  4 +       0
                → 4
```



COMPARATORS

Common Number Comparators used in JavaScript Syntax:

greater than

```
> 6 > 4
```

→ `true`

"boolean" value

less than

```
> 9 < 5
```

→ `false`

greater or equal

```
> 8 >= -2
```

→ `true`

equals

```
> 3 == 4
```

→ `false`

two equal signs!

not equals

```
> 12 != 4
```

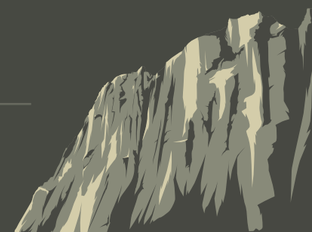
→ `true`

the "not" symbol

less or equal

```
> 10 <= 10
```

→ `true`



STRINGS

How JavaScript stores and processes flat text

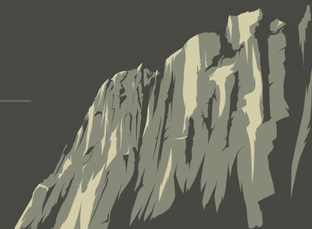
Strings need quotes!

```
> "Raindrops On Roses"
→ "Raindrops On Roses"
```

```
> "Whiskers On Kittens"
→ "Whiskers On Kittens"
```

Plus will glue Strings together

```
> "Raindrops On Roses" + " And " + "Whiskers On Kittens"
→ "Raindrops On Roses And Whiskers On Kittens"
```



THESE ARE A FEW OF MY FAVORITE...STRINGS

Concatenation works with numbers and their expressions, too.

```
> "The meaning of life is" + 42
```

→ "The meaning of life is42"

Uh oh...what happened? Concatenation adds no spaces, so we need to add our own.

Notice the extra space!

```
> "The meaning of life is " + 42
```

→ "The meaning of life is 42"

THESE ARE A FEW OF MY FAVORITE...STRINGS

Concatenation works with numbers and their expressions, too.

```
> "Platform " + 9 + " and " + 3/4
```

→ `"Platform 9 and 0.75"`

Expressions get evaluated!

```
>"Platform " + 9 + " and 3/4"
```

→ `"Platform 9 and 3/4"`

Make strings out of expressions that you want to see in their original format.

SPECIAL CHARACTERS INSIDE STRINGS

Some characters need backslash notation in JavaScript Strings

advances to the next "tab stop"

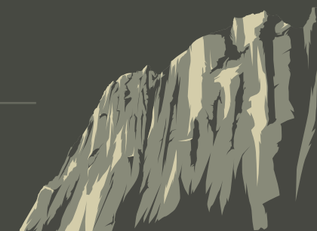
```
> "Flight #:\t921\t\tSeat:\t21C"
```

→ "Flight #: 921   Seat: 21C"

Adds a quotation mark but without ending the string too early.

```
> "Login Password:\t\t\"C3P0R2D2\""
```

→ "Login Password: "C3P0R2D2""



SPECIAL CHARACTERS INSIDE STRINGS

Some characters need backslash notation in JavaScript Strings

Places a backslash itself in the String

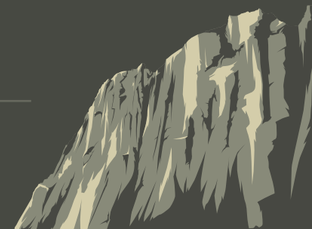
```
> "Origin\\Destination:\tOrlando(MCO)\\London(LHR)"
```

→ "Origin\Destination:	Orlando(MCO)\London(LHR)"

shifts the printout to a "new line"

```
> "Departure:\t09:55A\nArrival:\t14:55P"
```

→ "Departure:	09:55A
→ Arrival:	14:55P"



STRING COMPARISONS

Checking for matching strings and alphabetical ordering

"Double equals" will compare EXACT contents

```
> "The Wright Brothers" == "The Wright Brothers"
```

→ true

```
> "The Wright Brothers" == "Super Mario Brothers"
```

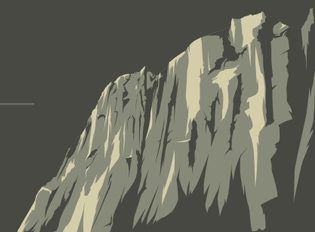
→ false

"Not equals" returns true if there is a mismatch

```
> "The Wright Brothers" != "the wright brothers"
```

→ true

Case counts!



STRING COMPARISONS

The length of strings can be accessed with the .length property

```
> "antidisestablishmentarianism".length
```

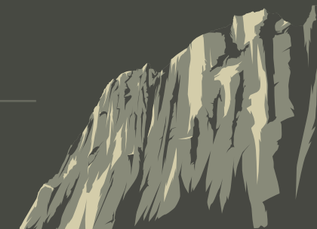
→ 28

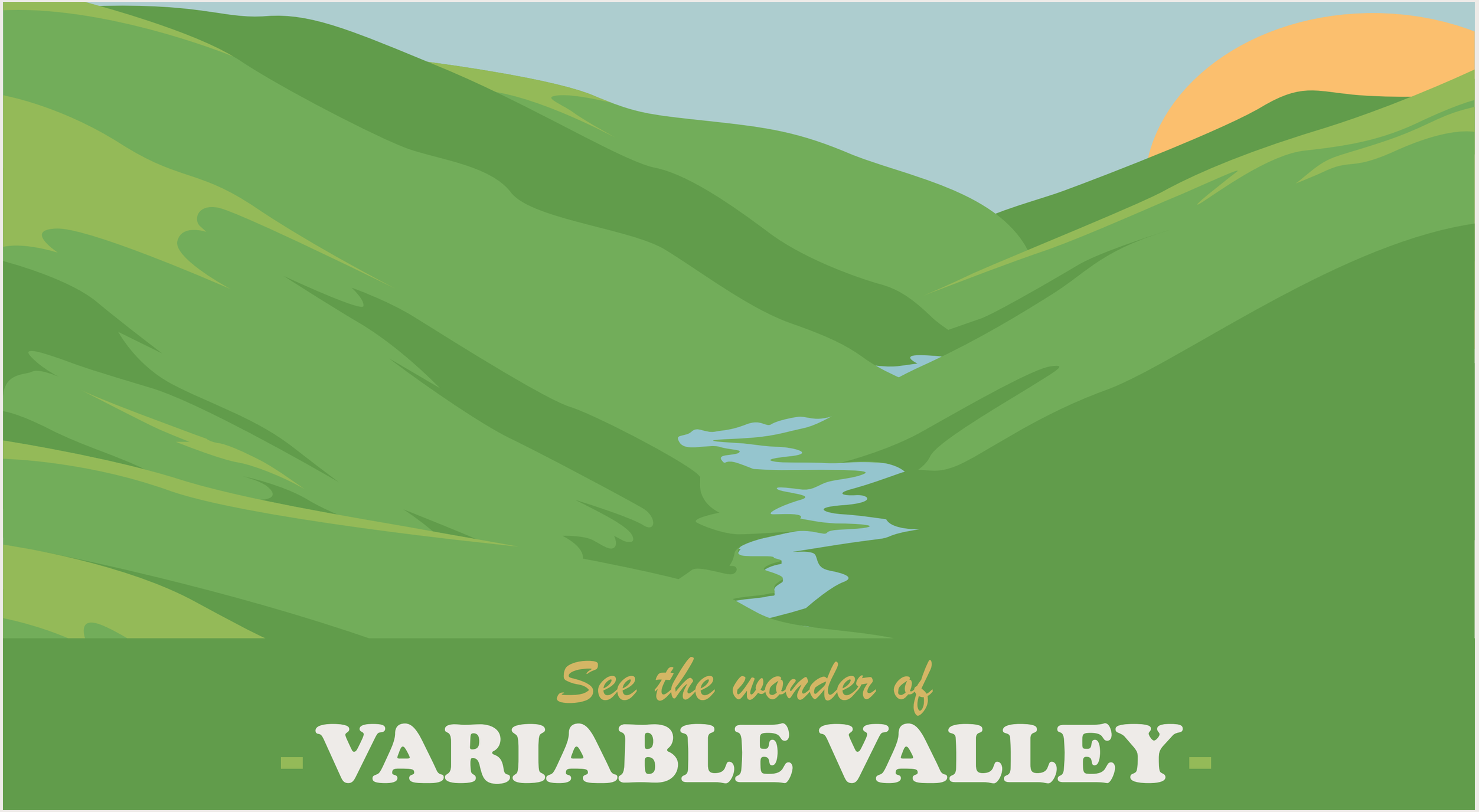
Returns a number value

Spaces and any non-alphabetic characters are counted, too!

```
> "One Fish, Two Fish, Red Fish, Blue Fish".length
```

→ 39



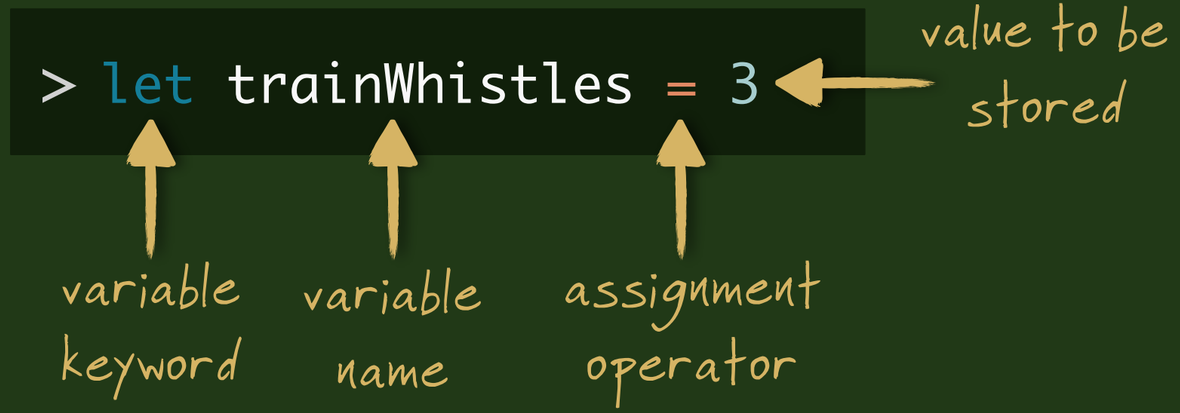

See the wonder of
VARIABLE VALLEY

LEVEL 2

VARIABLE VALLEY

STORING OUR VALUES

JavaScript uses variables to store and manage data



NAMING VARIABLES

Rules and regulations

`let no spaces`	❌	no spaces in the name
`let 3blindmice`	❌	no digits in front
`let scored_is_fine`	✅	underscores are okay, but often irritating
`let get$`	✅	dollar signs are also cool ... but don't ...
`let $_$`	✅	slightly stupid, but technically legal
`let goodNameHere`	✅	begin with lowercase, later words capitalized, "camel case"
`let mortalKombat2`	✅	FATALITY!!

CHANGING VARIABLE CONTENTS

Want to change a Variable's value? It's your lucky day.

```
>let trainWhistles = 3
```

```
> trainWhistles
```

→ 3

```
>trainWhistles = 9
```

no 'let' keyword this time, because JavaScript already "knows" about the variable

```
> trainWhistles
```

→ 9

```
>trainWhistles = trainWhistles + 3
```

uses current value to calculate new value

```
> trainWhistles
```

→ 12

CHANGING VARIABLE CONTENTS

Want to change a Variable's value? It's your lucky day.

```
> trainWhistles = trainWhistles + 3
```

same operation, different syntax

```
> trainWhistles += 3
```

```
> trainWhistles
```

→ 12

```
> trainWhistles
```

→ 15

CHANGING VARIABLE CONTENTS

Want to change a Variable's value? It's your lucky day.

```
> trainWhistles += 3
```

```
> trainWhistles
```

→ 15

```
> trainWhistles = trainWhistles * 2
```

same operation, different syntax

```
> trainWhistles
```

→ 30

```
> trainWhistles *= 2
```

```
> trainWhistles
```

→ 60

That's, like, a lot of whistles.

USING VARIABLES

Variable names also act as substitutes for the data they point to

```
> trainWhistles = 3
```

```
> "All of our trains have " + trainWhistles + " whistles!"
```

→ `"All of our trains have 3 whistles!"`

```
> "But the Pollack 9000 has " + (trainWhistles * 3) + "!"
```

→ `"But the Pollack 9000 has 9!"`

USING VARIABLES

Variable names also act as substitutes for the data they point to

```
> trainWhistles = 3
```

```
> let pollack9000 = trainWhistles * 3
```

```
> pollack9000
```

→ 9

USING VARIABLES

Variable names also act as substitutes for the data they point to

```
> trainWhistles = 3
```

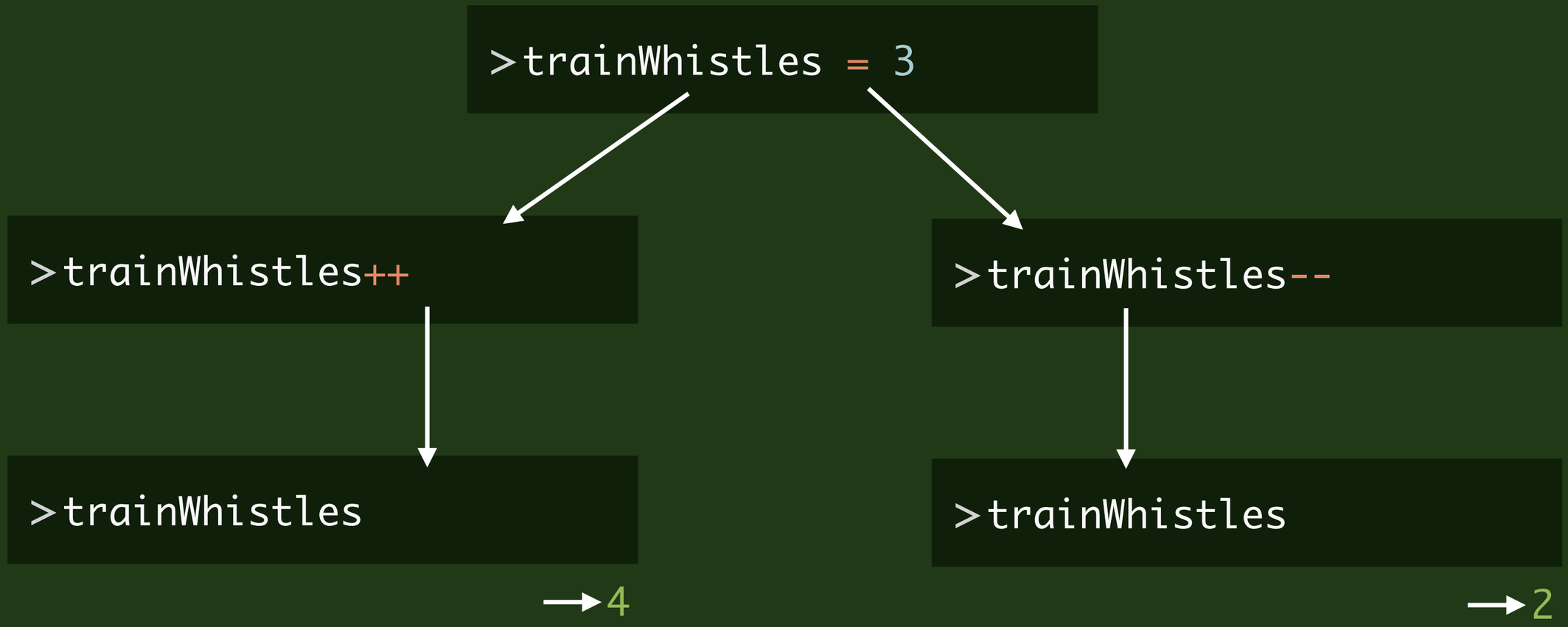
```
> let pollack9000 = trainWhistles * 3
```

```
> "But the Pollack 9000 has " + pollack9000 + "!"
```

→ `"But the Pollack 9000 has 9!"`

INCREMENTING AND DECREMENTING

A simple syntax for increasing or decreasing a variable's value by 1



VARIABLES STORE STRINGS, TOO!

JavaScript can store anything in variables.

```
> let welcome = "Welcome to the JavaScript Express Line!"
```

```
> let safetyTip = "Look both ways before crossing the tracks."
```

```
> welcome + "\n" + safetyTip
```

```
→ "Welcome to the JavaScript Express Line!
→ Look both ways before crossing the tracks."
```


USING VARIABLE NAMES WITH STRINGS

Variable names can also access the length property

```
> let longString = "I wouldn't want to retype this String every time."
```

```
> longString.length
```

→ 49

If a variable holds a String, we can access the length property directly from the variable name.

MORE COMPARISONS WITH VARIABLES

Comparing String lengths using the length property

```
> let longWordOne = "antidisestablishmentarianism"
```

```
> let longWordTwo = "supercalifragilisticexpialidocious"
```

Compares two numbers returned by the length properties

```
> longWordOne.length > longWordTwo.length
```

→ false

FINDING SPECIFIC CHARACTERS WITHIN STRINGS

Each position in a String has a numbered “index” starting from 0

```
> let sentence = "Antidisestablishmentarianism is fun to say!"
```


```
> sentence.length
43
```

Since the index starts at zero, but the length is counted by number of characters, the length value will always be one more than the last index.

FINDING SPECIFIC CHARACTERS WITHIN STRINGS

Each position in a String has a numbered “index” starting from 0

```
> let sentence = "Antidisestablishmentarianism is fun to say!"
```

```
> sentence.charAt(11)
```

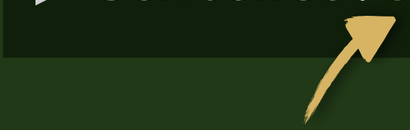
→ `"b"`

```
> sentence.charAt(31)
```

→ `" "`

```
> sentence.charAt(42)
```

→ `"!"`



The charAt( ) method retrieves the character at a specific index.

VARIABLES HELP ORGANIZE DATA

Creating a versatile message out of flexible pieces

```
> let trainsOperational = 8
```

```
> let totalTrains = 12
```

```
> let operatingStatus = " trains are operational today."
```

```
> trainsOperational + " out of " + totalTrains + operatingStatus
```

→ `"8 out of 12 trains are operational today."`

VARIABLES HELP ORGANIZE DATA

Creating a versatile message out of flexible pieces

```
> let trainsOperational = 10
```

```
> let totalTrains = 12
```

```
> let operatingStatus = " trains are operational today."
```

```
> trainsOperational + " out of " + totalTrains + operatingStatus
```

```
→ "10 out of 12 trains are operational today."
```