Lesson 01.02 - PROG

In this lesson:

- string concatenation with +
- string interpolation with ``
- numbers vs. "stringy numbers" (5 vs. "5")
- converting "stringy number" to number:
- Number(), parseInt(), parseFloat()
- "+" shortcut
- immutable vs. mutable
- NaN (Not a Number)
- math operators (besides +) work on stringy numbers
- not defined vs. undefined
- null (empty object)
- const for declaring constants

---

string concatenation with +

Variables and substrings can be joined together with + to make one bigger string. The procedure is known as "concatenation":

1. Concatenate a substring and a variable.
   Use a backslash \ to resolve inner quote conflicts:

```
let topic = 'JavaScript';
let intro = 'Let\'s learn ' + topic + '!';
console.log(intro); // Let's learn JavaScript!
```

2. Concatenate a substring with two variables:

```
let firstName = 'Brian';
let lastName = 'McClain';
let greeting = 'Hello! My name is ' + firstName + ' ' + lastName + '.';
console.log(greeting); // Hello, class! My name is Brian McClain.
```

3. Declare several variables and do a longer concatenation:

```
let petType = "cat";
let petName = 'Fluffy';
let age = 3;
let sound = 'Meow';
let catGreeting = sound + "! My name is " + petName + "! I am a " + age +
"-year-old " + petType + ".";
```

```
console.log(catGreeting); // Meow! My name is Fluffy! I am a 3-year-old
cat.
```

---

string interpolation with ``

This con-"cat"-enation has a ton of quotes and plus signs, making it hard to write. Fortunately, there is a simpler, cleaner alternative: "string interpolation." String interpolation has no quotes or plus signs: just one pair of backticks. To convert concatenation to interpolation, follow these steps:

- Delete all quotes and plus signs.
- Wrap everything in backticks .
- Wrap each variable in "dollar curlies": ${ } ${ } sets the variables apart from the hard-coded string parts.

4. Refactor the awkward con-"cat"-enation using string interpolation:

```
catGreeting = `${sound}! My name is ${petName}! I am a ${age}-year-old
${petType}.`;
console.log(catGreeting); // Meow! My name is Fluffy! I am a 3-year-old
cat.
```

---

numbers vs. "stringy numbers" (5 vs. "5")

A "stringy number" is a digit enclosed in quotes, such as "5".
Stringy numbers are strings, but with some number-like qualities.

5. Declare some actual numbers, as well as a stringy number.
   Then add them all together:

```
let foodAmt = 30;  // number
let bevAmt = 25; // number
let tip = '12.50';  // string
let tax = 7.75; // number
let bill = foodAmt + bevAmt + tip + tax;
console.log('bill:', bill); // bill: 5512.507.75
```

The result 5512.507.75 tells us that the first two numbers got added together,
but then it switched to concatenation when it encountered a "stringy number":

---

converting "stringy number" to number:

Number(), parseInt(), parseFloat()

- **Number()** converts its "stringy number" argument to an integer or float

- **parseInt()** converts it "stringy number" argument to an integer
- **parseFloat()** converts its "stringy number" argument to a float
  these all return NaN if passed a non-convertible string (like "banana")

6. Pass tip to the Number() method when adding up the bill. Now it works.

```
bill = foodAmt + bevAmt + Number(tip) + tax;
console.log('bill:', bill); // bill: 75.25
```

7. Try parseFloat() instead. This also works:

```
bill = foodAmt + bevAmt + parseFloat(tip) + tax;
console.log('bill:', bill); // bill: 75.25
```

8. Try parseInt(). This works--sort of. It drops the decimal,
   which knocks 50 cents off the tip:

```
bill = foodAmt + bevAmt + parseInt(tip) + tax;
console.log('bill:', bill); // bill: 74.75
```

- **+stringy_number** shortcut
- A plus sign in front of a stringy number works like Number(), parseInt(), parseFloat()

9. Declare a "stringy number", and then use + to get an actual number.

```
let price = '4.95';
price = +price;
console.log('+price:', price, typeof(price)); // +price: 4.95 number
```

10. Add up the bill again, using + to convert the string tip to a number:

```
bill = foodAmt + bevAmt + +tip + tax;
console.log('bill:', bill); // bill: 74.75
```

---

immutable vs. mutable

Above, we keep converting tip to a number, again and again.
This is because we aren't saving Number(tip), etc., to a variable

- numbers and strings are immutable (cannot be changed)

- Number(), parseInt(), +, etc. do not change their string argument
- methods return a copy which has the desired change
- set the Number(), etc. equal to a variable to save the return value
- the variable capturing the return value can be the same variable
- or it can be a new / different variable
- mutable datatypes include arrays and objects--a topic for another time

---

NaN (Not a Number)

NaN is the result of attempting an impossible numeric operation
If we try to get a number from a "banana", we just get NaN.
NaN stands for "Not a Number", and yet its datatype is number.

11. Try to convert "banana" to a number. We get NaN:

```
let baNaNa = Number('banana');
console.log('baNaNa:', baNaNa, typeof(baNaNa)); // baNaNa NaN number
```

12. Try that with the + shortcut. Once again, we get NaN:

```
baNaNa = +'banana';
console.log('baNaNa:', baNaNa, typeof(baNaNa)); // baNaNa NaN number
```

---

math operators (besides +) work on stringy numbers

- you can subtract, mutliply and divide with stringy numbers
- the *, - and / operators cannot be mistaken for the concatenation operator

13. Try subtracting, mutliplying and dividing with stringy numbers--it works!

```
// with stringy numbers:
console.log(tip + 10); // 12.5010 (addition fails)
console.log(tip * 10); // 125 (multiplication works)
console.log(tip - 10); // 2.5 (subtraction works)
console.log(tip / 10); // 1.25 (division works)
```

---

not defined vs. undefined

not defined is an error which means "variable does not exist".

- "not defined" arises from typos or other attempts to access variables which do not exist
- "not defined" is not to be confused with "undefined", which is not an error
- "undefined" is just a variable without a value

14. Declare a variable withoug assigning it a value
    log the variable, followed by a typo version:

```
let tot;
console.log('tot:', tot, typeof(tot)); // toy undefined undefined
// console.log(toy); // Error: toy is not defined
```

---

null (empty object)

- null is an empty object, with a data type of object.
- a variable can be intentionally set to null as a placeholder value
- or, a variable can be null as a result of a bug in our code.

15. Declare a null variable, and log it along with its datatype:

```
let user = null;
console.log('user:', user, typeof(user)); // user null object
```

---

const for declaring constants

- variables can change or *vary* -- hence the name
- constants cannot be changed -- hence the name
- constants, by convention, have UPPERCASE names
- but an all-caps name does not in itself make for a constant
- **const** keyword declares a constant that cannot be changed

16. Declare an uppercase variable with let:

```
let DOZEN = 12;
console.log(DOZEN); // 12
DOZEN = 13;
console.log(DOZEN); // 13
```

- despite being uppercase, DOZEN changed from 12 to 13,
- so the all-caps do not protect the value from changing.
- the all-caps merely advise programmers not to change the value.
- to declare a true constant -- that which cannot be changed -- use const
- if you try to change a **const**, you get an error:

17. Declare a true constant, and then try to change the value:

```
const PASSWORD = "abc";
// PASSWORD = "xyz"; // Error: Assignment to constant variable
```

DONE: Lesson 01.02

NEXT: Lab 01.02

LAB 01.02 - Solution

string concatenation with + and ""

string interpolation with `backticks`

1. Declare a variable called imgPath, and concatenate its values to be:
   "poker/images/cards/Jack-of-Hearts.png"
   Use the old-fashioned string concatenation with + and "":
   Use three variables in the concatenation:
   baseUrl is the folder path
   kind is "Jack"
   suit is "Hearts"

```
let baseUrl = "poker/images/cards/";
let kind = 'Jack';
let suit = 'Hearts';
let imgPath = baseUrl + kind + '-of-' + suit + '.png';
console.log(imgPath); // poker/images/cards/Jack-of-Hearts.jpg
```

2. Change variable values as required to get the Queen of Diamonds.
   But this time use the modern way: string interpolation `` (backticks):

```
kind = 'Queen';
suit = 'Diamonds';
imgPath = `${baseUrl} ${kind}-of-${suit}.png`;
console.log(imgPath); // poker/images/cards/Queen-of-Diamonds.png
```

3. Assign values to variables (employee, department).
   Use backticks to make the message: *Sally works in Sales.*

```
let employee = "Sally";
let department = "Sales";
let message = `${employee} works in ${department}.`;
console.log(message); // Sally works in Sales.
```

4. Assign values to variables (fName, country, months)
   Use backticks to make the message:
   Freddy lived in France for 8 months.

```
let fName = "Freddy";
let country = "France";
let months = 8;
message = `${fName} went to ${country} for ${months} months.`;
console.log(message); // Freddy lived in France for 8 months.
```

5. Assign values to variables (myPet, treat, meal)
   Use backticks to make the message:
   Bunny had blueberries for breakfast.

```
let myPet = "Bunny";
let treat = "blueberries";
let meal = "breakfast";
message = `${myPet} had ${treat} for ${meal}.`;
console.log(message); // Bunny had blueberries for breakfast.
```

6. Assign values to variables (dogName, toy)
   Use backticks to make the message:
   Fido’s favorite toy is his frisbee.

```
let dogName = 'Fido';
let toy = 'frisbee';
message = `${dogName}'s favorite toy is his ${toy}.`;
console.log(message); // Fido’s favorite toy is his frisbee.
```

mathy stuff

7. Change the value of one or more variables, so that instead // of getting NaN, we get the correct answer, which is 90.

```
let dayTripPrice = '$120';
const GROUPON = '30';
dayTripPrice = 120; // the one required change
let total = dayTripPrice - GROUPON;
console.log('total:', total); // 90
// stringy numbers work with subtraction, so remove just the '$'
// GROUPON cannot be changed, anyway, since it is a const
```

order of operations:

- exponents before multiplication and division
- multiplication and division before addition and subtraction

8. Calculate the total cost of 50 Kg (kilograms) of rice at $2.25 per Kg.
   Add 5% tax and $40 a delivery fee.
   Declare variables for ALL 4 numbers and one for the totalCost.

```
let kilosRice = 50;
let pricePerKg = 2.25;
let percentTax = 0.05;
let deliveryFee = 40;
let subTot = kilosRice * pricePerKg;
let totalCost = subTot + (subTot * percentTax) + deliveryFee;
console.log('totalCost:', totalCost);
```

9. Reproduce the JAVASCRIPT CAFE GUEST CHECK as shown.
   Declare variables for each individual piece of data:
   placeName, foodTot, bevTot, etc. etc.
   Make the message useing string interpolation, with backticks
   Inside backticks, Enter and Tab work as expected.
   (Enter and Tab do not work with regular concatenation.)

   JAVASCRIPT CAFE GUEST CHECK: Food tot: $30 Bev tot: $20 Sub-tot: $50 Tax pct: 8% Tax tot: $4 Tip pct: 20% Tip tot: $10 PLEASE PAY: $64 THANK YOU! */

```
let placeName = 'JavaScript Cafe';
let foodTot = 30;
let bevTot = 20;
let subTotal = foodTot + bevTot;
let taxPct = 0.08; // 8%
let tipPct = 0.2; // 20%;
let taxTot = subTotal * taxPct;
let tipTot = subTotal * tipPct;
let grandTotal = subTotal + taxTot + tipTot;

let guestCheck = `
    ${placeName}
    GUEST CHECK:
        Food tot: $${foodTot}
        Bev tot: $${bevTot}
        Sub-tot: $${subTotal}
        Tax pct: ${taxPct * 100}%
        Tax tot: $${taxTot}
        Tip pct: ${tipPct * 100}%
        Tip tot: $${taxTot}
        PLEASE PAY: $${grandTotal}
    THANK YOU!`;

console.log(guestCheck);
```

// END 01.02 LAB

THEN: Lesson 01.03