Date Object

Outputting date and time to the web page

The Date Object returns the full date and time from the user's computer. It is instantiated (declared) using the **new** keyword.

1. Instantiate an instance of the Date object.

```
let dateTime = new Date();
console.log(dateTime);
```

The individual time units are available by calling the Date object's "get methods".

2. Get the current hour, minute and second:

```
let hour = dateTime.getHours();
console.log(hour); // 0-23

let minute = dateTime.getMinutes();
console.log(minute); // 0-59

let second = dateTime.getSeconds();
console.log(second); // 0-59
```

3. Express the time in 00:00:00 format:

```
let timeIs = `${hour}:${minute}:${second}`;
console.log(timeIs);
```

leading zeroes for minute and second

If minute or second is less than 10, you get wonky output, such as 1:2:3, instead of 1:02:03. To fix this, add leading zeros to minute and second, as needed. This is done with conditional logic.

4. Add a leading 0 if minute or second is less than 10:

```
if(minute < 10) {
  minute = '0' + minute;
}

if(second < 10) {
  second = '0' + second;
}

timeIs = `${hour}:${minute}:${second}`;
console.log(timeIs);
```

If an if-statement has only one line of code inside its curly braces, you can omit the curly braces altogether and put everything on the same line.

5. Make these short if-statements even more concise by eliminating the curly braces:

```
if(minute < 10) minute = '0' + minute;
if(second < 10) second = '0' + second;

timeIs = `${hour}:${minute}:${second}`;

console.log('timeIs w leading 0', timeIs); // 00:00:00
```

converting military time to AM/PM

The hour is from 0-23 ("military time"), so 3pm is 15:00 and 10pm is 22:00.

To convert to AM/PM time, we need:

- a variable to store the string "AM" or "PM".
- two if-statements, done in this order:
  - if hour > 11, use "PM".
  - if hour > than 12, subtract 12.

6. Declare a variable **amOrPm** with an initial value of 'AM', and follow that with the if-statements:

```
let amOrPm = 'AM';

if(hour > 11) {
    amOrPm = 'PM';
}

if(hour > 12) {
    hour -= 12;
}
```

```
    timeIs = `${hour}:${minute}:${second} ${amOrPm}`;
    console.log('time is: ', timeIs);
```

timely greeting

Now let's make a "timely greeting" that is appropriate for the current hour:

- **if** the hour is less than 12 (noon), say "Good morning!".
- **else if** the hour is less than 18 (6:00pm), say "Good afternoon!"
- **else**, say "Good Evening!"

Start the greeting with "Good" and then use += to concatenate the "timely" part:

7. Get a fresh hour, since our original hour may have already had 12 subtracted from it:

```
    let hr = dateTime.getHours();
```

8. Declare greeting with an initial value of 'Good':

```
    let greeting = "Good ";
```

9. Do the logic for **hr < 12** (noon):

```
    if(hr < 12) {
        greeting = "morning";
    }
```

10. Add an **else if** for **hr < 18** (6pm). Follow that with an **else** that runs when **hr** is 18 and up:

```
    if (hr < 12) {
      greeting += "Morning!";
    } else if (hr < 18) {
      greeting += "Afternoon!";
    } else {
      greeting += "Evening!";
    }

    console.log(greeting);
```

"Timely Greeting"

11. Pair the greeting with the time of day in AM-PM format:

```
```

```
let timelyGreeting = `${greeting} The time is: ${timeAMPM}`;
console.log(timelyGreeting);
```

12. Output the "timely greeting" to the web page. Start by getting the tag that will display the greeting:

```
let greetingTag = document.getElementById('greeting');
```

13. Set the **textContent** property of the tag object to **timelyGreeting**:

```
greetingTag.textContent = timelyGreeting;
```

The Date object's other time units can be used to concatenate and output today's date:

14. Get today's date:

```
let date = dateTime.getDate();
console.log('date', date);
```

15. Get the month, which is returned as a number from 0-11 (Jan = 0, Dec = 11):

```
let month = dateTime.getMonth();
console.log('month', month);
```

16. Get the month as a string (January, February, etc). This gives us the flexibility to use the month as either a number or a day:

```
let fullMonth = dateTime.toLocaleString('default', {month:'long'})
console.log('fullMonth', fullMonth);
```

17. Get the day of the week, which is a number, with Sunday=0 and Saturday=6:

```
let day = dateTime.getDay();
console.log('day', day);
```

18. Make an array of the days of the week.

```
let daysArr = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"];
```

We'll explore arrays thoroughly in upcoming lessons, but basically, an array is a variable that stores multiple values as a list, inside square brackets.

19. Look up the first item in the array (Sunday) by its index (0):

```
let Sun = daysArr[0];
console.log('Sun', Sun); // Sun Sunday
```

20. Get the day of the week by looking it up index (number) in the array:

```
let dayOfWeek = daysArr[day];
console.log('dayOfWeek', dayOfWeek); // string
```

21. Get the full 4-digit year:

```
let fullYear = dateTime.getFullYear();
console.log(fullYear);
```

22. Concatenate today's date as **Day, Month Date, Year**, using the non-numeric day and month, e.g. **Tuesday, May 17, 2022**:

```
let todaysDate = `${dayOfWeek}, ${fullMonth} ${date}, ${fullYear}`;
console.log('todaysDate', todaysDate); // string
```

23. Output today's date to its place on the web page:

```
let todaysDateTag = document.getElementById('todays-date');

todaysDateTag.innerHTML = todaysDate;
```

updating the time every second

You may be wondering: Why doesn't the time on the web page updating every 1 second? To auto-update every second would require more code. In a later lesson, we'll get into using **setInterval** to call a function X times per second to do just this sort of thing.

END Lesson 02.04

NO LAB EXERCISES

PROCEED DIRECTLY TO LESSON 03.01