LESSON 6
DATA TYPES II

The Nature of Life Is to Grow

Wholeness: JavaScript is a rapidly evolving language. This lesson covers several new data type features that make the language more efficient and powerful. *Science of Consciousness:* The nature of life is to grow and evolve to greater accomplishment and fulfillment.

Main Points

1. Destructuring assignment
2. Date and time
3. JSON

Main Point Preview: Destructuring assignments

Destructuring assignment is a new (ES6) and now widely used convenience syntax that allows us to “unpack” arrays or objects into a set of variables.
Science of Consciousness: This feature allows developers to more quickly accomplish common coding tasks. Do less and accomplish more.

Destructuring assignment

- special syntax that allows us to “unpack” arrays or objects into a set of variables

```
let arr = ["Ilya", "Kantor"]

// sets firstName = arr[0] and surname = arr[1]
let [firstName, surname] = arr;
```

- It works great when combined with split or other array-returning methods:

```
let [firstName, surname] = "Ilya Kantor".split(' ‘);
```

Does not change the array.

‘Syntactic sugar’ to replace the following:

```
let firstName = arr[0];
let surname = arr[1];
```

Destructuring assignment 2

Unwanted elements of the array can also be thrown away via an extra comma:

```
// second element is not needed
let [firstName, , title] = ["Julius", "Caesar", "Consul", "of the Roman Republic"];
alert( title ); // Consul
```

can use any “assignables” at the left side.

```
let user = {};
[user.name, user.surname] = "Ilya Kantor".split(' ');
alert(user.name); // Ilya
```

Destructuring assignment -- …rest

- The rest '…'
  - can add one more parameter that gets "the rest" using three dots "...":
  - value of rest is array of remaining elements.
  - can use any other variable name in place of rest
    - three dots before it
    - last in the destructuring assignment.

```
let [name1, name2, ...rest] = ["Julius", "Caesar", "Consul", "of the Roman Republic"];
alert(name1); // Julius
alert(name2); // Caesar

// Note that type of `rest` is Array.
alert(rest[0]); // Consul
alert(rest[1]); // of the Roman Republic
alert(rest.length); // 2
```

Destructuring assignment -- missing and default values

- Default values
  - If different number of values in array than variables in assignment, there will be no error.
  - Absent values are considered undefined:
  - Extra values are ignored

```
let [firstName, surname] = [];
alert(firstName); // undefined
alert(surname); // undefined
```

- If we want a “default” value to replace the missing one, we can provide it using =:

```
// default values
let [name = "Guest", surname = "Anonymous"] = ["Julius"];
alert(name);    // Julius (from array)
alert(surname); // Anonymous (default used)
```

Exercise

```
const team = [ "Bob",  "Fred",  "Jim“]
// destructure the team array onto variables with the same names as the properties, but all lower case



console.log("expect Bob", bob );
console.log("expect Jim", jim);
```

Object destructuring

- destructuring assignment also works with objects

```
let options = {
  title: "Menu",
  width: 100,
  height: 200
};
let {title, width, height} = options;
alert(title);  // Menu
alert(width);  // 100
alert(height); // 200
```

- Properties are assigned to the corresponding variables.
  - order does not matter

```
let {height, width, title} = { title: "Menu", height: 200, width: 100 }
```

Destructure property to another name

- to assign a property to a variable with another name, set it using a colon

```
let options = {
  title: "Menu",
  width: 100,
  height: 200
};

// { sourceProperty: targetVariable }
let {width: w, height: h, title} = options;

// width -> w
// height -> h
// title -> title

alert(title);  // Menu
alert(w);      // 100
alert(h);      // 200
```

wrap destructuring expression in parentheses (---)

- use existing variables without let. there's a catch.

```
let title, width, height;

// error in this line
{title, width, height} = {title: "Menu", width: 200, height: 100};
```

- JavaScript assumes a code block instead of destructuring

- To show it's not a code block, wrap the expression in parentheses (---):

```
// okay now
({title, width, height} = {title: "Menu", width: 200, height: 100});
```

Object destructuring – default values and parameters

- For potentially missing properties can set default values using "="

```
let options = {
  title: "Menu"
};
let {width = 100, height = 200, title} = options;
alert(title);  // Menu
alert(width);  // 100
alert(height); // 200
```

- can pass parameters as an object, and the function destructures them into parameters:

```
let options = {
  title: "My menu",
  items: ["Item1", "Item2"]
};
function showMenu({title = "Untitled", width = 200, height = 100, items = []}) {
  // title, items – taken from options,
  // width, height – defaults used
  alert( `${title} ${width} ${height}` ); // My Menu 200 100
  alert( items ); // Item1, Item2
}
showMenu(options);
```

Exercise

```
const team = { point: "Bob", shooting: "Fred", power: "Jim", small: "Al",  center:  "Big Sleep" }

// 1. destructure the team object onto variables with the same names as the properties



console.log("expect Big Sleep", center);
console.log("expect Jim", power);



// 2. destructure the team onto variables:  one (point guard), two (shooting guard), three (small forward), four (power forward) and five (center)



console.log("expect Jim: ", four);
console.log("expect Bob: ", one);
```

destructuring exercises

- Destructuring assignment

- The maximal salary
  - use built-in method Object.entries(obj)
    - returns array of key/value pairs for an object

```
for (let [key, value] of Object.entries(object1))
  {  console.log(`${key}: ${value}`);}
```

Main Point : Destructuring assignments

Destructuring assignment is a new (ES6) and now widely used convenience syntax that allows us to “unpack” arrays or objects into a set of variables.
Science of Consciousness: This feature allows developers to more quickly accomplish common coding tasks. Do less and accomplish more.

Main Point : Date and time

JavaScript has added a new Date object has many convenient methods for working with subtle aspects of dates and time. *Science of Consciousness:* Dates are important in many common programming tasks and involve complex logic. This is an important development for the language. The nature of life is to grow and evolve.

Date and time

- a new built-in object: Date
  - can store creation/modification times, measure time, or print out current date

- Without arguments – create a Date object for the current date and time:

```
let now = new Date();
console.log( now ); // shows current date/time, e.g., Thu Sep 12 2019 12:49:12 GMT-0500 (Central Daylight Time)

let date = new Date("2017-01-26");
console.log(date);
// timezone not set, so assumed midnight GMT and adjusted according to the timezone the code is run in
// So the result could be
// Thu Jan 26 2017 11:00:00 GMT+1100 (Australian Eastern Daylight Time)
// or
// Wed Jan 25 2017 16:00:00 GMT-0800 (Pacific Standard Time)
```

*

Access/set date components

getFullYear()
getMonth() // month, from 0 to 11.
getDate() //day of month, from 1 to 31
getDay() //day of week, from 0 (Sunday) to 6 (Saturday).
getHours(), getMinutes(), getSeconds(), getMilliseconds()
getTime() //number milliseconds from January 1st of 1970 UTC+0.

setFullYear(year, [month], [date])
setMonth(month, [date])
setDate(date)
setHours(hour, [min], [sec], [ms])
setMinutes(min, [sec], [ms])

Date to number, date diff

When a Date is converted to number, it becomes the timestamp as with date.getTime():

```
let start = new Date(); // start measuring time
// do the job
for (let i = 0; i < 100000; i++) {
  let doSomething = i * i * i;
}
let end = new Date(); // end measuring time
console.log( `The loop took ${end - start} ms` );  // can be used for time measurements
```

- If we only want to measure time, there's a special method Date.now() that returns the current timestamp
  - equivalent to new Date().getTime(), but it doesn't create an intermediate Date object.
  - faster and doesn't put pressure on garbage collection.

```
let start = Date.now(); // milliseconds count from 1 Jan 1970
// do the job
for (let i = 0; i < 100000; i++) {
  let doSomething = i * i * i;
}
let end = Date.now(); // done
console.log( `The loop took ${end - start} ms` ); // subtract numbers, not dates
```

Date.parse from a string

- read a date from a string.
- format should be: YYYY-MM-DDTHH:mm:ss.sssZ, where:
  - YYYY-MM-DD – is the date: year-month-day.
  - character "T" is used as the delimiter.
  - HH:mm:ss.sss – is the time: hours, minutes, seconds and milliseconds.
  - optional 'Z' part denotes time zone.
  - Shorter variants are also possible, like YYYY-MM-DD or YYYY-MM or even YYYY.

- Date.parse(str) returns the timestamp
  - number of milliseconds from 1 Jan 1970 UTC+0). If the format is invalid, returns NaN.

```
let ms = Date.parse('2013-01-26T18:00:00.417-21:00');  //6PM in Ethiopia
alert(ms); // 1359298800417  (timestamp)
//create a new Date object from the timestamp:
let date = new Date(ms);
alert(date);  //Sun Jan 27 2013 09:00:00 GMT-0600 (Central Standard Time)
```

Exercise

Write code that measures how long it takes to find the maximal subarray for an array that has 10 elements and another that has 100 elements using first the O(n) solution and then the O(n**2) solution.

Homework exercises

- Create a date
- Show a weekday (implement in vscode with tests)
- European weekday
- Which day of month was many days ago?
  - Hint: date.setDate(0); // min day is 1, so the last day of the previous month is assumed
- How many seconds has passed today?

Main Point: Date and time

JavaScript has added a new Date object has many convenient methods for working with subtle aspects of dates and time. *Science of Consciousness:* Dates are important in many common programming tasks and involve complex logic. This is an important development for the language. The nature of life is to grow and evolve.

Main Point Preview: JSON

JSON is a widely used data exchange format used in every modern programming language. It is based on JavaScript object representations. The main uses are to convert JavaScript objects to strings for transmission over a network and to parse JSON strings sent from servers into JavaScript objects. *Science of Consciousness:* JSON is a mechanism for improving communication between remote entities—clients and servers. Coherent and orderly consciousness is a mechanism for improving communication between people.

JSON methods, toJSON

- Often must convert objects into strings
  - send it over a network,
  - output it for logging purposes.

```
let user = {
  name: "John",
  age: 30,
  toString:  function() {
    return `{name: "${this.name}", age: ${this.age}}`;
  }
};
alert(user); // {name: "John", age: 30}
```

- …But new properties are added, old properties are renamed and removed.
  - Updating such toString every time can become a pain.

JSON.stringify

- JSON (JavaScript Object Notation) is a general format to represent values and objects.
  - RFC 4627 standard.
  - Initially for JavaScript, but many other languages have libraries to handle it as well.
  - data exchange when client uses JavaScript and the server uses Ruby/PHP/Java/Whatever.

JSON.stringify to convert objects into JSON.
JSON.parse to convert JSON back into an object

```
let student = {
  name: 'John',
  age: 30,
  isAdmin: false,
  courses: ['html', 'css', 'js'],
  wife: null};

let json = JSON.stringify(student);
```

```
{  "name": "John",
  "age": 30,
  "isAdmin": false,
  "courses": ["html", "css", "js"],
  "wife": null}
```

- important differences from the object literal:
  - Strings use double quotes.
  - No single quotes or backticks in JSON.
  - Object property names double-quoted
- JSON is data-only
- some object properties are skipped
  - Function properties (methods).
  - Symbolic properties.
  - Properties that store undefined.

JSON.parse

- convert JSON back into an object

```
// stringified array
let numbers = "[0, 1, 2, 3]";
numbers = JSON.parse(numbers);
alert( numbers[1] ); // 1

let user = '{ "name": "John", "age": 35, "isAdmin": false, "friends": [0,1,2,3] }';
user = JSON.parse(user);
alert( user.friends[1] ); // 1
```

typical mistakes in hand-written JSON

```
let json = `{
  name: "John",                     // mistake: property name without quotes
  "surname": 'Smith',               // mistake: single quotes in value (must be double)
  'isAdmin': false                  // mistake: single quotes in key (must be double)
  "birthday": new Date(2000, 2, 3), // mistake: no "new" is allowed, only bare values
  "friends": [0,1,2,3]              // here all fine
}`;
```

- **Exercises**
  - Turn the object into JSON and back
    - Can this be used for cloning objects?

Main Point : JSON

JSON is a widely used data exchange format used in every modern programming language. It is based on JavaScript object representations. The main uses are to convert JavaScripts to strings for transmission over a network and to parse JSON strings sent from servers into JavaScript objects. *Science of Consciousness:* JSON is a mechanism for improving communication between remote entities—clients and servers. Coherent and orderly consciousness is a mechanism for improving communication between people.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

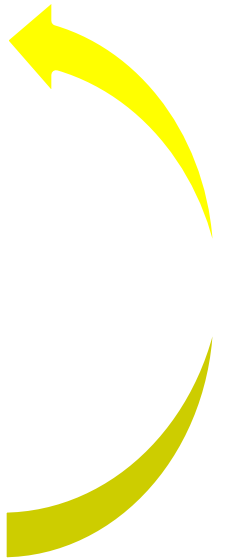
The Nature of Life Is to Grow

1. Destructuring assignments, the Date object, and JSON are new features and data representations in JavaScript.

2. Destructuring provides syntactic convenience and programming efficiency, Date provides capabilities, and JSON enables remote transfer of data.

---

3. **Transcendental consciousness**. Is the simplest state of awareness and home of all the laws of nature.
4. **Impulses within the transcendental field:** Thoughts at this level will be maximally life supporting.
5. **Wholeness moving within itself:** In unity consciousness one experiences the goal of all growth and evolution in daily life.



JavaScript Object Notation (JSON)

JSON is a syntax for storing and exchanging data and an efficient alternative to XML

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

A name/value pair consists of a field name **(in double quotes)**, followed by a colon, followed by a value.

JSON values can be:

- A number (integer or floating point)
- A string (in double quotes)
- A Boolean (true or false)
- An array (in square brackets)
- An object (in curly braces)
- null

JSON expressions exercise

Given the JSON data at right, what expressions would produce:

- The window's title?
- The image's third coordinate?
- The number of messages?
- The y-offset of the last message?

```
const jsonString = '{
  "window": {
    "title": "Sample Widget",
    "width": 500,
    "height": 500
  },
  "image": {
    "src": "images/logo.png",
    "coords": [250, 150, 350, 400],
    "alignment": "center"
  },
  "messages": [
    {"text": "Save", "offset": [10, 30]}
    {"text": "Help", "offset": [ 0, 50]},
    {"text": "Quit", "offset": [30, 10]},
  ],
  "debug": "true"
}';
const data = JSON.parse(jsonString);
```