LearnFrenzy

Basics of JavaScript for Salesforce Developers

1. What is JavaScript for LWC?

- JavaScript controls the behavior of your Lightning Web Components.
  It is the brain behind your component — handling logic, user interactions, data processing, APEX calls, and more.
- JavaScript is a scripting or programming language that allows you to implement complex features on web pages.
- It brings interactivity to the web pages, helps component logics and server communication.
- Mastering modern JS (ES6+) is critical for building dynamic, responsive, and high-performing Salesforce components.

Master these concepts in JavaScript learning path:

1. Variables
2. Data Types
3. null VS undefined
4. Arrow functions
5. Spread Operator
6. Destructuring
7. String Interpolation
8. String Methods
9. Object methods
10. Array Methods
11. Promises
12. Modules import and export
13. Events
14. setTimeout vs setInterval
15. QuerySelectors

Now, Let's get started….

1. Variables:

- A variable is like a container that holds data you can use and manipulate in your component.
- Var, const and let are the reserved keyword to declare a variable.
- JavaScript identifiers area case sensitive.
- You can assign a value to a variable using equal to (=) operator when you declare it or before using it.
  Keyword level scope:

Keyword	const	let	var
Global scope	Yes	Yes	Yes
Function scope	Yes	Yes	Yes
Block scope	Yes	Yes	No
Can be reassigned	No	Yes	Yes

- let variable can be updated but not re-declared.
- const variables can neither be updated nor re-declared.

Basics of JavaScript for Salesforce Developers

2. Data Types:

- A data type defines the kind of value a variable can hold — such as numbers, text, Booleans (true/false), objects, etc.
- In Salesforce LWC development, you deal with JavaScript data types all the time — handling user input, API responses, Apex method results, form fields, and more.

Data types have 2 main categories: Primitive Datatypes
Non-Primitive Datatypes

Primitive Datatypes:

Data Type	Example	Description
String	"Salesforce", 'LWC'	Sequence of characters (text)
Number	100, 3.14, -50	Integer or floating-point number
Boolean	true, false	Logical true/false
Undefined	let a; (a is undefined)	Variable declared but no value assigned
Null	let b = null;	Explicitly no value
BigInt	1234567890123456789012345678901234567890n	Large integers (rarely needed in LWC)
Symbol	Symbol('desc')	Unique and immutable value (advanced usage)

Non-Primitive Datatypes:

Data Type	Example	Description
Object	{ name: 'Salesforce', type: 'CRM' }	Collection of key-value pairs
Array	[10, 20, 30]	Ordered list of values
Function	function greet() {}	Blocks of reusable code
Class	class Account {}	Blueprint for creating objects (used in OOP patterns)

3. Null vs Undefined

Null: - null is a special assignment value.
- It means you intentionally assigned "no value" to a variable.
- It represents nothing, but you explicitly decided that.

Undefined: - undefined means a variable is declared, but no value is assigned yet.
- It's the default value for uninitialized variables.

Basics of JavaScript for Salesforce Developers

S.NO	undefined	null
1	If a variable is declared, but not initialized or assigned any value, then JS automatically initializes it with undefined.	It's a special data type that represents "nothing", "empty", or "value unknown". It is assigned explicitly.
2	typeof undefined is undefined	typeof null is object
3	undefined == null is true	undefined === null is false

Equality Comparison:
== This operator does value comparison. Example 100 == “100”
=== this operator does value plus data type comparison. Example 100 === “100”

4. Spread Operator:

- The spread operator (…) is used to expand or "spread" elements of an array or object into individual elements.
- In Salesforce LWC, it is heavily used to update objects immutably, clone arrays, merge objects, and pass parameters.

Usages of spread operator –
Expanding String – Convert string into list of arrays
Combining Arrays – Combine array or add value to array
Combining Object – Combine Object or add value to Object
Creating New Shallow Copy of Arrays and objects

5. Destructuring:

- Destructuring in JavaScript allows you to unpack values from arrays or properties from objects into separate variables — easily and cleanly.
- In Salesforce LWC, destructuring makes Apex data handling, event handling, and state management faster and cleaner.

Type	Example	Where Useful
Array Destructuring	Extract multiple values from arrays	Lists of data, picklists
Object Destructuring	Extract properties from objects	Apex responses, event details

6. String Interpolation:

- String Interpolation allows you to embed expressions in the string.
- Template strings use back-ticks (``) rather than the single or double quotes.

In Salesforce Lightning Web Components (LWC), string interpolation is widely used to:

- Display dynamic content
- Generate API URLs
- Create error messages
- Customize UI labels

```
let name = 'Salesforce';
let message = `Welcome to ${name}!`;

console.log(message);
// Output: Welcome to Salesforce!
```

7. String Methods:

- String methods are built-in JavaScript functions that help you manipulate, search, format, or analyze string data.
- In Salesforce development (LWC, Aura, Visualforce), string methods are crucial when you:
  - Handle API responses
  - Build dynamic labels/messages
  - Process Apex returned data
  - Manage user inputs

Common String Methods:

Method	Purpose	Example
length	Get the length of a string	'Hello'.length // 5
toUpperCase()	Convert to uppercase	'salesforce'.toUpperCase() → 'SALESFORCE'
toLowerCase()	Convert to lowercase	'LWC'.toLowerCase() → 'lwc'
includes()	Check if string contains text	'Salesforce'.includes('force') // true
startsWith()	Check if string starts with text	'Hello World'.startsWith('Hello') // true
endsWith()	Check if string ends with text	'data.csv'.endsWith('.csv') // true
indexOf()	Find the first index of a substring	'Lightning'.indexOf('n') // 5
substring()	Extract part of a string	'Salesforce'.substring(0,5) → 'Sales'
slice()	Extract portion of string	'Salesforce'.slice(5) → 'force'
replace()	Replace part of a string	'LWC Rocks'.replace('Rocks', 'Rules') → 'LWC Rules'
split()	Split string into array	'a,b,c'.split(',') → ['a','b','c']
trim()	Remove whitespace from both ends	' Salesforce '.trim() → 'Salesforce'
repeat()	Repeat a string	'LWC '.repeat(3) → 'LWC LWC LWC '

8. Object/JSON Operations:

- Object: A key-value pair collection, where the key is always a string, and the value can be any data type.
- JSON (JavaScript Object Notation): A string representation of a JavaScript object — used for data transmission between systems (like Salesforce Apex ↔ LWC).

  In Salesforce LWC, you deal with objects when:

- Fetching data from Apex controllers
- Handling API responses
- Updating Salesforce records
- Passing data between components

Most frequently used operations:

a. **Object.keys()** - it extracts just the keys (field names) of an object.
b. **Object.values()** - it extracts just the values (not the keys).
c. **JSON.strigify()** - converts a JavaScript object or array into a JSON-formatted string
d. **JSON.parse()** - takes a JSON-formatted string and converts it back into a JavaScript object or array.

9. Array Methods:

Most important array methods:

Method	Purpose	Example
push()	Add an item at the end	arr.push('newItem')
pop()	Remove the last item	arr.pop()
shift()	Remove the first item	arr.shift()
unshift()	Add an item at the beginning	arr.unshift('newItem')
splice()	Add/Remove items at any position	arr.splice(2, 1, 'newItem')
slice()	Copy part of the array (non-destructive)	arr.slice(1, 3)
map()	Transform array into a new array	arr.map(x => x*2)
filter()	Filter array based on condition	arr.filter(x => x > 5)
find()	Find the first matching element	arr.find(x => x > 5)
forEach()	Execute a function for each element	arr.forEach(x => console.log(x))
reduce()	Reduce array to a single value	arr.reduce((sum, x) => sum + x, 0)
includes()	Check if item exists	arr.includes('value')
indexOf()	Find index of an item	arr.indexOf('value')
sort()	Sort array elements	arr.sort()

10. Promise:

- A Promise is a special JavaScript object that represents the eventual completion (or failure) of an asynchronous operation and its resulting value.
- In other words, promise is an object that may produce a single value sometime in then future.
- Promise has 3 states
    i. pending()
    ii. fulfilled()
    iii. rejected()

Use case from LWC point of view:

1. Fetching data from server
2. Loading file from system

11. Modules Imports and Exports :

Exports:

Exporting - use export keyword to export many variable or many method from a file

```
export const name = "Kaushik"

export function getName(){
  return "Kaushik"
}
```

Default export – use export default keyword to export only one variable or a method from a file.

```
export default user = "salesforce"
```

Imports:

Importing – use import keyword to import variable or method from a given file path or module.

Multiple imports

```
import { name, getname } from './filepath'
```

Imports all exported members

```
import * as Utils from './filepath'
```

Imports a module with a default member

```
import user from './filepath'
```

12. QuerySelector:

The querySelector () method returns the first element that matches a specified CSS selector(s) in the document.

```
document.querySelector(selector);
```

querySelectorAll

The querySelectorAll () method returns all elements in the document that matches a specified CSS selector(s), as a static nodeList object.

```
document.querySelectorAll(selector);
```

13. Events:

An event is an action that occurs in the web browser, which the web browser feedbacks to you so that you can respond to it.

For example, when users click a button on a webpage, you may want to respond to this click event by displaying a alert box.

Event handler - It is a block of code that will execute when the event occurs. It is also known as an event listener.

Two Common ways to add events

1. HTML Event Handler attribute - When we add event through HTML, Event always begin with on keyword like onclick, onchange, onkeyup.

2. Event Listener - Event Handlers provide two main methods for dealing with the registering/deregistering event listeners:
    - **addEventListener()** – register an event handler
    - **removeEventListener()** - remove an event handler

Event Propagation

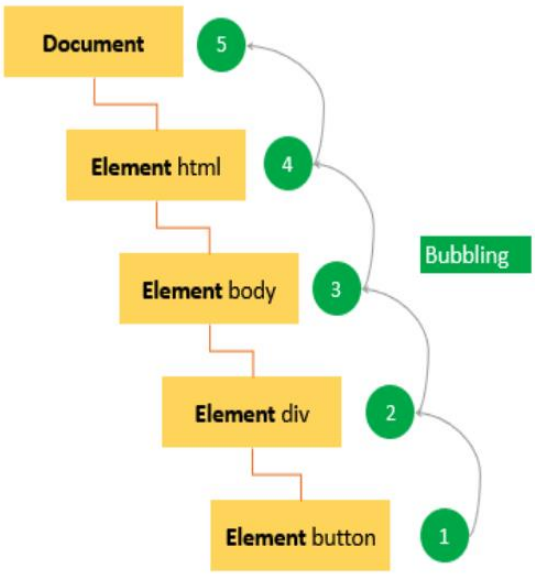
Event Propagation explains the order in which events are received on the page from the element where the event occurs and propagated through the DOM tree.

There are two main event models

i. Event bubbling
ii. Event Capturing

Event Bubbling:

In the event bubbling model, an event starts at the most specific element and then flows upward toward the least specific element (the document or even window).



Custom Event:

A Custom Event is an event created manually by a developer to send specific information from a child component to a parent component.

Syntax:

```
new CustomEvent("eventName", {options})
```

Example:

```
const event = new CustomEvent('eventname', {
    detail: {  note:"Hello People!" },
    bubbles: true,
    composed: true
});
this.dispatchEvent(event);
```

14. Arrow functions:

- An arrow function (=>) is a shorter and cleaner way to write functions in JavaScript.

Example:

```
const greet = (name) => `Hello, ${name}`;
```

Benefits of Arrow functions:

- Arrow syntax automatically binds this to the surrounding code's context.
- Prefer arrow functions for cleaner, safer, and faster coding — especially inside Promise chains, event handlers, and asynchronous operations.

15. setTimeout vs setInterval:

setTimeout

The setTimeout) is a method of the window object. The set Timeout () sets a timer and executes a callback function after the timer expires.

Syntax:

```
setTimeout(function, delayInMilliseconds);
```

setInterval

The setInterval) is a method of the window object. The setInterval) repeatedly calls a function with a fixed delay between each call.

Syntax:

```
setInterval(function, intervalInMilliseconds);
```