

# JS MODERN CONCEPTS



# 1. Optional Chaining (?.)

Introduced in ECMAScript 2020, optional chaining allows you to read the value of a property located deep within a chain of connected objects without having to check that each reference in the chain is valid.

*OptionalChining.js*

```
let name = person?.address?.street?.name;
```

# 2. Nullish Coalescing (??)

Also introduced in ECMAScript 2020, the nullish coalescing operator returns the first operand if it's not null or undefined, and the second operand otherwise.

*Nullish Coalescing.js*

```
let name = person?.name ?? 'Unknown' ;
```



# 3. BigInt

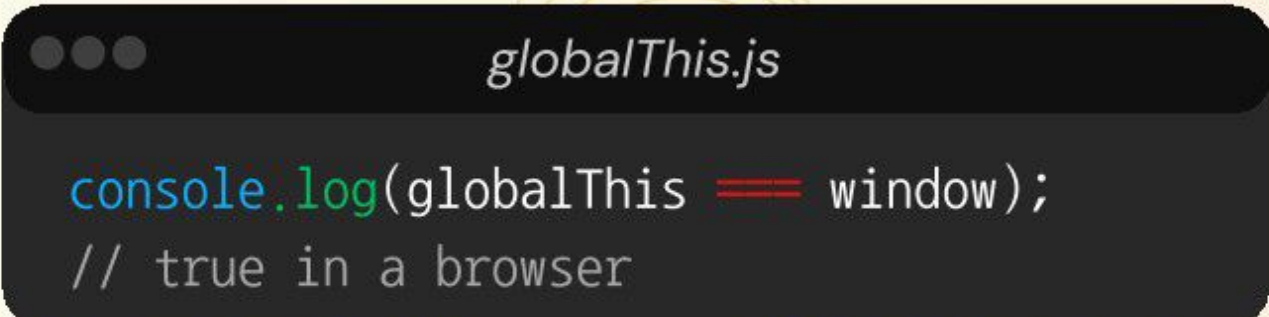
A new numeric primitive in JavaScript, BigInt is used to represent integers with arbitrary precision, allowing for accurate calculations with large integers.

*BigInt.js*

```
const x = 12345678901234567890n;
```

# 4.globalThis

A new global object, `globalThis`, provides a way to access the global object in a way that's compatible with modern JavaScript environments.

A dark-themed code editor window with a title bar containing three window control buttons and the filename 'globalThis.js'. The editor contains two lines of JavaScript code: 'console.log(globalThis === window);' and a comment '// true in a browser'.

```
globalThis.js  
  
console.log(globalThis === window);  
// true in a browser
```

# 5. `matchAll()`

A new method on the String prototype, `matchAll()` returns an iterator that yields matches of a regular expression against a string, including capturing groups.

```
matchAll.js

const regex = /(\w)(\d)/g;

const str = 'a1b2c3';

for (const match of str.matchAll(regex)) {
  console.log(match);
}
```

# 6. Promise.allSettled()

A new method on the Promise API, `allSettled()` returns a promise that is resolved when all of the promises in an array are either resolved or rejected.

*Promise.allSettled.js*

```
const promises = [Promise.resolve('a'), Promise.reject('b'),  
Promise.resolve('c')];  
Promise.allSettled(promises).then((results) =>  
console.log(results));
```



# 7. **String.prototype.at()**

A new method on the String prototype, `at()` returns the character at the specified index, allowing for negative indices to access characters from the end of the string.

```
String.prototype.at.js

const str = 'hello';

console.log(str.at(0)); // 'h'
console.log(str.at(-1)); // 'o'
```



# 8. Error Cause

A new property on Error objects, `cause` allows you to specify the underlying cause of an error.

*Promise.allSettled.js*

```
try {  
  throw new Error("Error occurred", { cause: new Error("Underlying  
cause") }); } catch (error) {  
  console.log(error.cause);  
}
```