

# Taller de Node - Sesión 2

Javascript - ES6 - File System - Read Line

RONY HANCCO

GIRO PINTO

# Javascript data types

```
let variable ---> undefined
```

```
let variable = null ---> null
```


```
let stringVariable = 'javascript' ---> string
```

```
let numberVariable = 123 ---> number
```

```
let booleanVariable = true ---> boolean
```

```
let objectVariable = {} ---> object
```

```
let symVariable = Symbol() ---> symbol
```



**JS**

**Data Types in  
JavaScript**

# Variables

```
var variableName
```

```
let variableName
```

```
const variableName
```

# VAR vs LET vs CONST

	var	let	const
Stored in Global Scope	✓	✗	✗
Function Scope	✓	✓	✓
Block Scope	✗	✓	✓
Can Be Reassigned?	✓	✓	✗
Can Be Redeclared?	✓	✗	✗
Can Be Hoisted?	✓	✗	✗

# Arreglos

```
const arr = ['a', 'b', 'c']
```

# Objetos

```
const object = {  
  prop1: 'hola',  
  prop2: 123,  
  prop3: [],  
  prop4: {},  
  ...  
}
```

# Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Operator	Description
&&	logical and
	logical or
!	logical not

# If-Else if -Else

```
if (num > 15) {  
  return 'Mayor que 15'  
} else if (num < 5) {  
  return 'Menor que 5'  
} else {  
  return 'Entre 5 y 15'  
}
```

# Operador Ternario

```
return a > b ? 'a es el mayor' : 'b es el mayor'
```



# Switch

```
switch(num) {  
  case value1:  
    //...  
    break;  
  case value2:  
    //...  
    break;  
  case value3:  
    //...  
    break;  
}
```



# While

```
let i = 0

while (i < 10) {
  console.log(i)
  i++
}
```

# Do while

```
let i = 0

do {
  console.log(i)
  i++
} while (i < 5)
```

# For loop

```
const arr = ['a', 'b', 'c']

for (let i = 0; i < arr.length; ++i) {
  console.log(arr[i])
}
```

# For of

```
const arr = ['a', 'b', 'c']

for (let v of arr) {
  console.log(v)
}
```

# forEach

```
const arr = ['a', 'b', 'c']

arr.forEach((v, i) => console.log(v))
```

# For in

```
const object = { a: 1, b: 2, c: 3 }

for (const i in object) {
  console.log(`${i}: ${object[i]}`)
}
```

# Functions

```
function consoleLogSomething() {  
  console.log('Something')  
}  
  
const consoleLogSomething = function () {  
  console.log('Something')  
}
```

# Arrow functions

```
const consoleLogSomething = () => {  
  console.log('Something')  
}  
  
const returnValue = () => 'value'
```

# Import

```
import React , { useContext } from 'react'
```

# Export

```
export { myClass, someFunction , ... }
```

# Spread Operator

```
const arr = ['a', 'b', 'c']
```

```
let arr2 = [...arr, 'd']
```

```
const defaults = {  
  author: '',  
  title: '',  
  year: 2021,  
  rating: 5,  
}
```

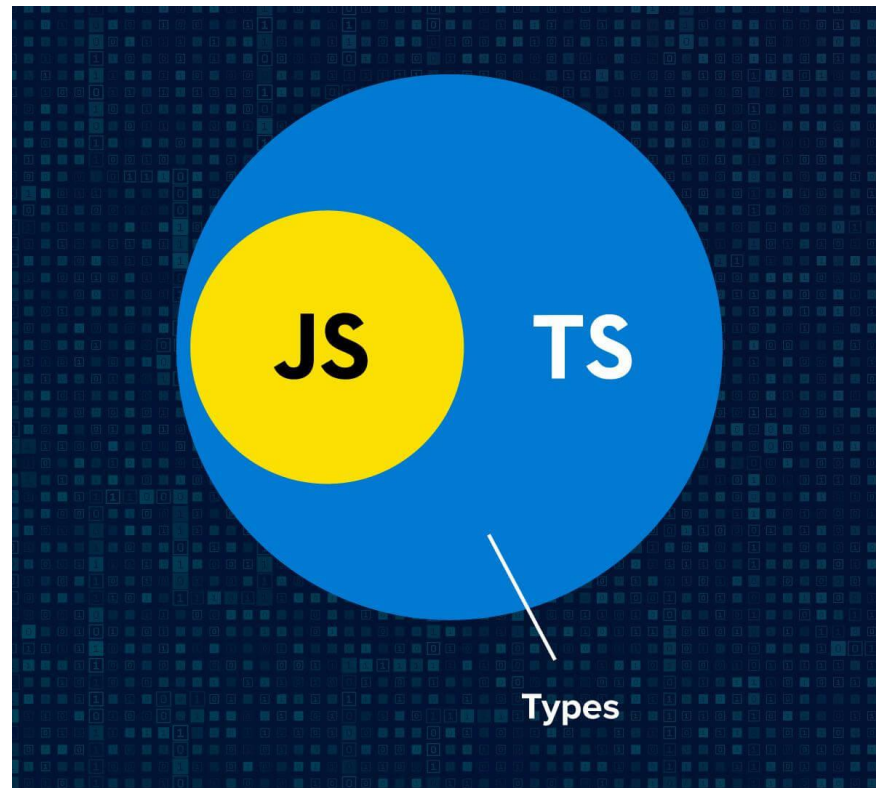
```
const book = {  
  author: 'Eloquent Javascript',  
  title: 'Marijn Haverbeke',  
}
```

```
const bookWithDefaults = { ...defaults, ...book }
```

# Object destructuring

```
const book = {  
  
  author: 'Eloquent Javascript',  
  title: 'Marijn Haverbeke',  
  rating: 5  
}  
  
const {author as bookAuthor, year='2018'} = book  
  
const {  
  author,  
  ...additional  
} = book
```

# Next Steps



A diagram illustrating the JavaScript Event Loop. It features a large 'JS' logo on a yellow background. A circular arrow labeled 'EVENT LOOP' is shown on a dark background. The text 'Handle asynchronous actions' is present. Below it, the following code snippets are shown:

```
Promise()  
  .then(data => {})  
  .catch(err => {})  
  
async () => { await }
```

## Object-Oriented Programming in Javascript (ES6)



Abstraction | Inheritance | Encapsulation | Polymorphism



# File System

ReadFiles

WriteFiles



ReadLine

Read Line doc.