

Security

SQL Injection



Cosa è:

SQL injection è una tecnica di *code injection* dove si inietta del codice SQL

```
# Define POST variables
uname = request.POST['username']
passwd = request.POST['password']

# SQL query vulnerable to SQLi
sql = "SELECT id FROM users WHERE username='" + uname + "' AND password='" + passwd + "'"

# Execute the SQL statement
database.execute(sql)
```

<https://www.acunetix.com/websitesecurity/sql-injection/>

Come si combatte?

Semplicemente usando: **prepared statements and parameterized queries**

```
$stmt = $dbConnection->prepare('SELECT * FROM employees WHERE name = ?');  
$stmt->bind_param('s', $name);
```

Oppure pulendo tutti gli input:

```
mysqli_real_escape_string ( mysqli $link , string $escapestr ) : string
```

This function is used to create a legal SQL string that you can use in an SQL statement. The given string is encoded to an escaped SQL string, taking into account the current character set of the connection.

```
$unsafe_variable = $_POST["user-input"];  
$safe_variable = mysqli_real_escape_string($unsafe_variable);  
mysql_query("INSERT INTO table (column) VALUES (" . $safe_variable . ")");
```

Javascript



JavaScript was initially created to “make web pages alive”.

Scripts are provided and executed as plain text. They don't need special preparation or compilation to run.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Statements</h2>

<p>A <b>JavaScript program</b> is a list of <b>statements</b> to be executed by a computer.</p>

<p id="demo"></p>

<script>
var x, y, z; // Declare 3 variables
x = 5;      // Assign the value 5 to x
y = 6;      // Assign the value 6 to y
z = x + y;  // Assign the sum of x and y to z

document.getElementById("demo").innerHTML =
"The value of z is " + z + ".";
</script>

</body>
</html>
```

https://www.w3schools.com/js/js_examples.asp

Javascript

Javascript è un linguaggio debolmente tipizzato

<https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/>



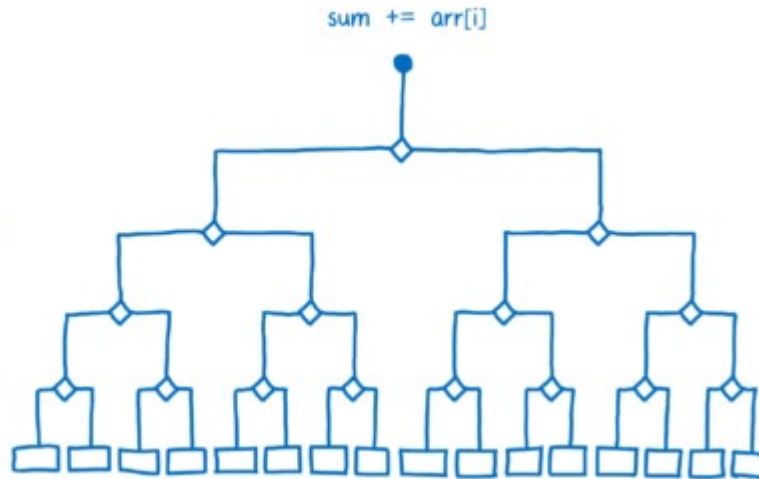
```
function arraySum(arr) {  
  var sum = 0;  
  for (var i = 0; i < arr.length; i++) {  
    sum += arr[i];  
  }  
}
```

is sum an int?

is arr an array?

is i an int?

is arr[i] an int?



Javascript

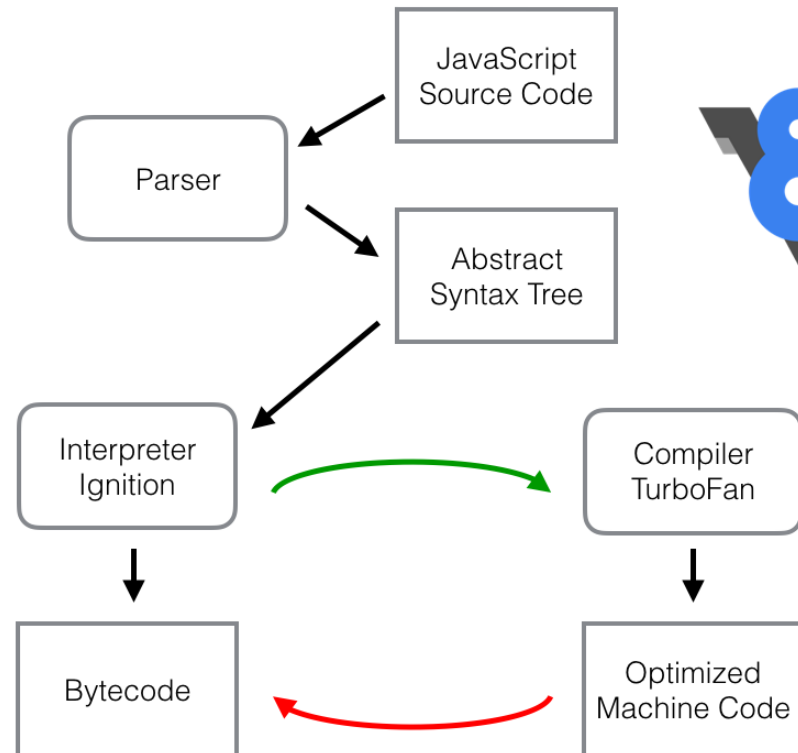


(SpiderMonkey)

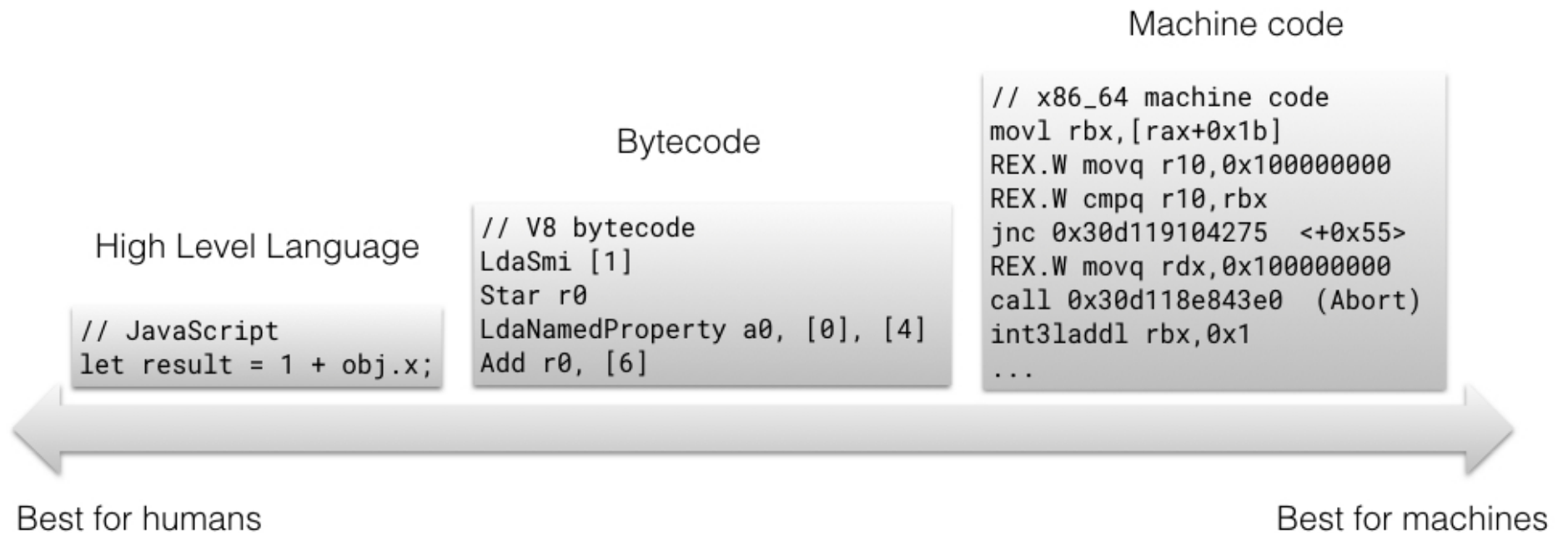


(Nitro)

1. The engine (embedded if it's a browser) reads ("parses") the script.
2. Then it converts ("compiles") the script to the machine language.
3. And then the machine code runs, pretty fast.



Javascript



@fhinkel

Javascript

JavaScript is always synchronous and single-threaded. If you're executing a JavaScript block of code on a page then no other JavaScript on that page will currently be executed.



synchronous, single thread of control



synchronous, two threads of control



asynchronous



Javascript – Callback and Promise

One approach to asynchronous programming is to make functions that perform a slow action take an extra argument, a *callback function*. The action is started, and when it finishes, the callback function is called with the result.

```
setTimeout(() => console.log("Tick"), 500);
```

A *promise* is an asynchronous action that may complete at some point and produce a value. It is able to notify anyone who is interested when its value is available.

```
let fifteen = Promise.resolve(15);  
fifteen.then(value => console.log(`Got ${value}`));
```



Javascript – Callback and Promise

Javascript Program

```
function baz(){  
  console.log('Hello from baz');  
}  
  
function bar() {  
  baz();  
}  
  
function foo() {  
  bar();  
}  
  
foo();
```

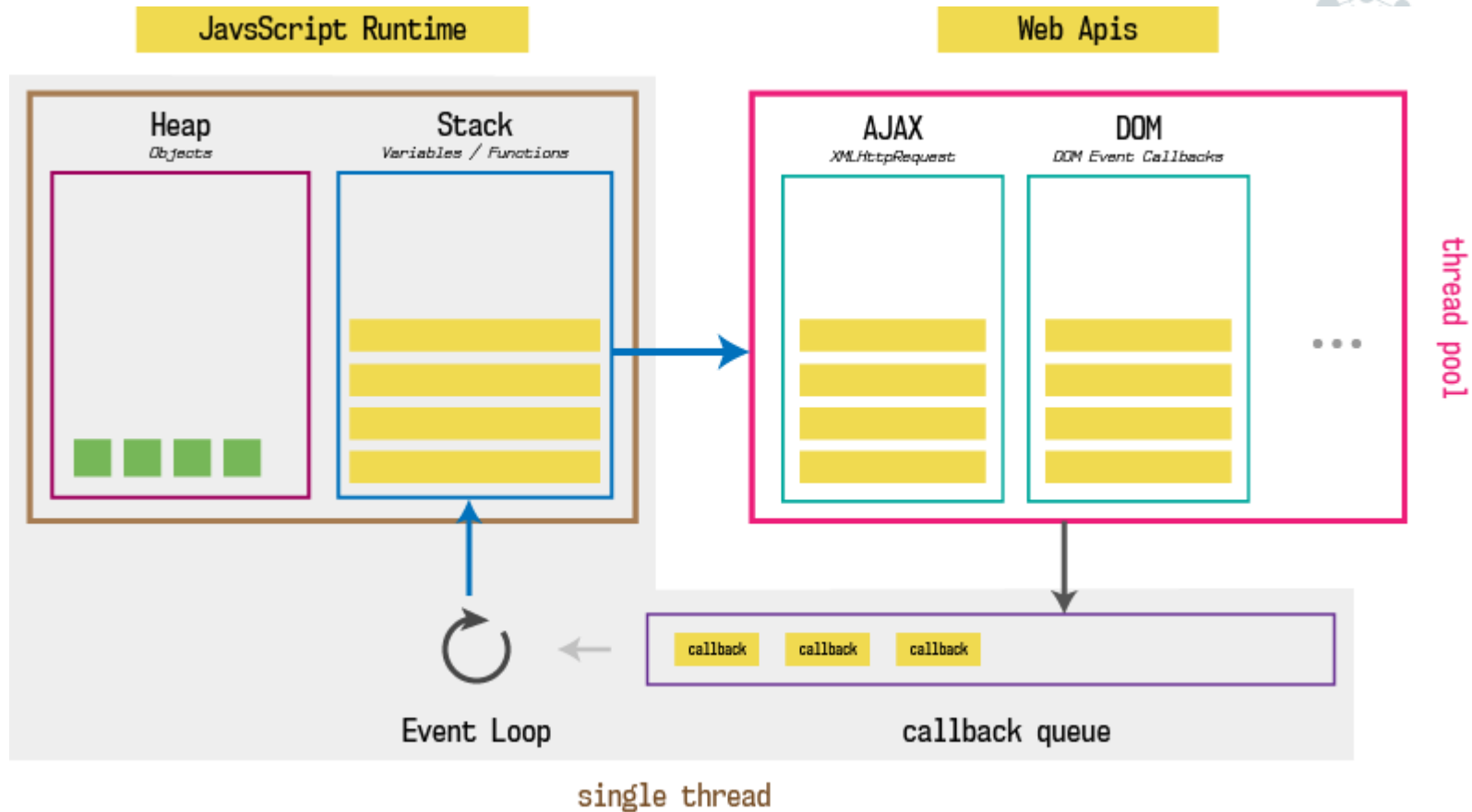
Javascript Runtime

Heap
Objects

Stack
Variables / Functions

foo()

Javascript – Callback and Promise



Javascript – Callback and Promise

The screenshot shows the Loupe JavaScript playground interface. The top bar is orange with the Loupe logo and a 'help' link. The left panel contains a code editor with the following JavaScript code:

```
1  
2  
3 function printHello() {  
4   console.log('Hello from baz');  
5 }  
6  
7 function baz() {  
8   setTimeout(printHello, 3000);  
9 }  
10  
11 function bar() {  
12   baz();  
13 }  
14  
15 function foo() {  
16   bar();  
17 }  
18  
19 foo();
```

Below the code editor is a 'Click me!' button and an 'Edit' button. The right panel is divided into three sections: 'Call Stack', 'Web Apis', and 'Callback Queue'. The 'Call Stack' and 'Web Apis' sections are empty. The 'Callback Queue' section contains a single blue circle icon, representing a callback function waiting to be executed. A red circular arrow icon is positioned between the 'Call Stack' and 'Web Apis' sections and the 'Callback Queue' section, indicating the flow of execution from the call stack to the callback queue.

Javascript – Callback and Promise

<https://www.youtube.com/watch?v=8aGhZQkoFbQ>

