Presented by

CSE-23-16 CSE-23-28

CSE-23-55 CSE-23-68

An open-source and *cross-platform JavaScript runtime* environment.

An open-source and *cross-platform* *JavaScript* *runtime* environment.

JavaScript runtime

Is simply a computer program that executes JavaScript code. It's responsible for translating human-readable JavaScript code into machine-readable instructions that the computer's hardware can execute.

JavaScript *Mental Model*

Process

```
→ node index.js
```

JavaScript *Mental Model*

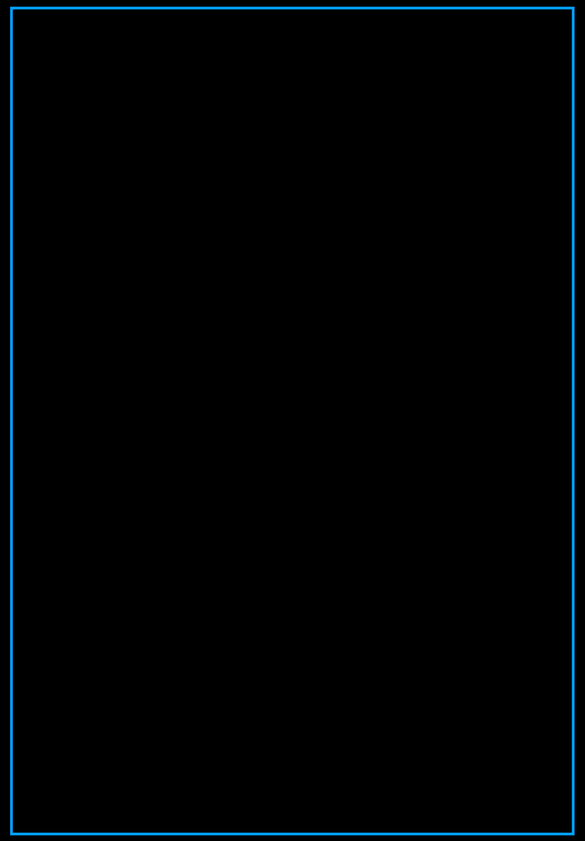
Process

callstack

```
→ node index.js
```

JavaScript *Mental Model*

Process



callstack

```
→ node index.js
```

what is to be executed has to be at top of stack

Process

callstack

Scope

process

callstack

Scope

```
const num = 3;
const a = 'some string';

function getStatus() {
  const status = 'up'
  return status;
}

console.log(getStatus());
console.log(status);
```

console

process

callstack

Scope

```
const num = 3;
const a = 'some string';

function getStatus() {
  const status = 'up'
  return status;
}

console.log(getStatus());
console.log(status);
```

up

console

process

callstack

Scope

```
const num = 3;
const a = 'some string';

function getStatus() {
  const status = 'up'
  return status;
}

console.log(getStatus());
console.log(status);
```

```
undefined
```

console

process

callstack

Global
Scope

```
const num = 3;
const a = 'some string';

function getStatus() {
  const status = 'up'
  return status;
}

console.log(getStatus());
console.log(status);
```

console

process

callstack

```
const num = 3;
const a = 'some string';

function getStatus() {
  const status = 'up'
  return status;
}

console.log(getStatus());
console.log(status);
```

process

callstack

Thread of *Execution*

Global (scope)

```javascript
var num = 3;
var a = 'some string';

function getStatus() {
  console.log(status)
  var status = 'up' + a;
  return status;
}

console.log(getStatus());
console.log(status);
console.log(num);
```

callstack

Thread of *Execution*

Global *(scope)*

```
num: undefined;
a: undefined;
getStatus: function {}
```

>

console

callstack

Thread of *Execution*

Global *(scope)*

```
num: undefined;
a: undefined;
getStatus: function {}
```

```
var num = 3;
var a = 'some string';

function getStatus() {
  console.log(status)
  var status = 'up' + a;
  return status;
}
```

>

console

callstack

Thread of *Execution*

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(
  getStatus()
);
```

```
console.log(status);

console.log(a);
```

>

console

callstack

getStatus

Thread of *Execution*

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(
  getStatus()
);
```

```
console.log(status);

console.log(a);
```

>

console

callstack

Thread of *Execution*

getStatus

Body of function	local memory

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(
  getStatus()
);
```

```
console.log(status);

console.log(a);
```

>

console

callstack

Thread of *Execution*

getStatus

Body of function

```
console.log(status);

const status = 'up' + a;

return status;
```

local memory *(scope)*

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(
  getStatus()
);
```

```
console.log(status);

console.log(a);
```

>

console

callstack

Thread of *Execution*

getStatus

Body of function

```
console.log(status);

const status = 'up' + a;

return status;
```

local memory *(scope)*

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(
  getStatus()
);
```

```
console.log(status);

console.log(a);
```

```
> Cannot access 'status' before initialization
```

console

let

const

var

function

let

const

```
// using
console.log(a);

const a = 3;

//using
console.log(a);
```

let

const

Temporal *Dead Zone*

```
// using
console.log(a);

const a = 3;

//using
console.log(a);
```

```
> ReferenceError: Cannot access 'a' before initialization
```

console

let

const

```
// using
console.log(a);
```

```
const a = 3;

//using
console.log(a);
```

```
>
> 3
```

console

let

const

```
const x = 1;
{
  console.log(x);
  const x = 2;
}
```

```
>
```

console

let

const

```
const x = 1;
{
  console.log(x);
  const x = 2;
}
```

```
> ReferenceError: Cannot access ‘x' before initialization
```

console

var

```
// using
console.log(a);

var a = 3;

//using
console.log(a);
```

```
> undefined
```

console

var

```
// using
console.log(a);

var a = 3;

//using
console.log(a);
```

```
> undefined
> 3
```

console

var

```
{
  var x = 1;
}
console.log(x); // 1
```

```
> 1
>
```

console

function

```
>
```

console

function

async function

*function **

*async function **

function

```
>
```

console

```
// using
write();

function write() {
  console.log('JavaScript is dope xP')
}

// using
write();
```

function

```
>
```

console

```
// using
write();
```

```
function write() {
  console.log('JavaScript is dope xP')
}

// using
write();
```

function

```
> JavaScript is dope xP
```

console

```
// using
write();

function write() {
  console.log('JavaScript is dope xP')
}
```

function

```
// using
write();
```

```
>
> JavaScript is dope xP
```

console

callstack

Thread of *Execution*

getStatus

Body of function

```
console.log(status)

const status = 'up'

return status;
```

local memory

```
status: 'up'
```

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(
  getStatus()
);
```

```
console.log(status);

console.log(a);
```

```
> Cannot access 'status' before initialization
```

console

callstack

Thread of *Execution*

getStatus

Body of function

```
console.log(status)

const status = 'up'

return status;
```

local memory

status: 'up'

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(
  getStatus()
);
```

```
console.log(status);

console.log(a);
```

>

console

callstack

Thread of *Execution*

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(
    'up'
);
```

```
console.log(status);

console.log(a);
```

>

console

callstack

console.log

Thread of *Execution*

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(status);

console.log(a);
```

>

console

callstack

Thread of *Execution*

console.log

Body of function

```
if (onLog.hasSubscribers) {
  onLog.publish(args);
}
this[kWriteToConsole](
  kUseStdout,
  this[kFormatForStdout](args)
);
```

local memory *(scope)*

```
args: ['up']
```

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(status);
```

```
console.log(a);
```

>

console

callstack

Thread of *Execution*

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(status);

console.log(a);
```

```
> up some string
>
```

console

callstack

Thread of *Execution*

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(status);
```

```
console.log(a);
```

```
>
> undefined
```

console

callstack

Thread of *Execution*

Global *(scope)*

```
num: 3;
a: 'some strng'
getStatus: function {}
```

```
console.log(status);
```

```
console.log(a);
```

```
>
> some string
```

console

```
function getStatusFunction() {
  const status = 'up';
  function getStatus() {
    return status;
  }
  return getStatus;
}

const getStatus = getStatusFunction();
console.log(getStatus())
```

```
function getStatusFunction() {
  const status = 'up';
  function getStatus() {
    return status;
  }
  return getStatus;
}

const getStatus = getStatusFunction();
console.log(getStatus())
```

```
> up
```

console

callstack

Thread of *Execution*

Global *(scope)*

```
getStatusFunction:
        function {}
```

```
const getStatus
  = getStatusFunction();
```

```
console.log(getStatus())
```

>

console

callstack

getStatusFunction

Thread of *Execution*

Global *(scope)*

```
getStatusFunction:
        function {}
```

```
const getStatus
  = getStatusFunction();
```

```
console.log(getStatus())
```

>

console

callstack

Thread of *Execution*

getStatusFunction

Body of function

```
const status = 'up';
function getStatus()
  return status;
}
return getStatus;
```

local memory *(scope)*

```
status: 'up'
getStatus:
 function {}
```

Global *(scope)*

```
getStatusFunction:
        function {}
```

```
const getStatus
  = getStatusFunction();
```

```
console.log(getStatus())
```

>

console

callstack

Thread of *Execution*

getStatusFunction

Body of function

```
const status = 'up';
function getStatus()
  return status;
}
return getStatus;
```

local memory *(scope)*

```
status: 'up'
getStatus:
 function {}
```

Global *(scope)*

```
getStatusFunction:
        function {}


const getStatus
  = getStatusFunction();



console.log(getStatus())
```

>

console

callstack

Thread of *Execution*

Global *(scope)*

```
getStatusFunction:
        function {}
```

```
getStatus
function {}
```


```
const getStatus
  = getStatusFunction();


console.log(getStatus())
```

>

console

callstack

getStatus

Thread of *Execution*

Global *(scope)*

```
getStatusFunction:
        function {}
```

```
getStatus
function {}
```


```
const getStatus
  = getStatusFunction();
```

```
console.log(getStatus())
```

>

console

callstack

Thread of *Execution*

getStatus

Body of function

```
return status;
```

local memory *(scope)*

Global *(scope)*

```
getStatusFunction:
        function {}
```

```
getStatus
function {}
```


```
const getStatus
  = getStatusFunction();
```

```
console.log(getStatus())
```

>

console

callstack

Thread of *Execution*

getStatus

Body of function

```
return status;
```

local memory
(scope)

Global *(scope)*

```
getStatusFunction:
        function {}
```

```
getStatus
function {}
```


```
const getStatus
  = getStatusFunction();

console.log(getStatus())
```

>

console

callstack

Thread of *Execution*

getStatus

Body of function

```
return status;
```

local memory
(scope)

Closure

Global *(scope)*

```
getStatusFunction:
        function {}
```

```
getStatus
function {}
```


```
const getStatus
  = getStatusFunction();


console.log(getStatus())
```

>

console

callstack

Thread of *Execution*

Global *(scope)*

```
getStatusFunction:
        function {}

getStatus
function {}
```


```
const getStatus
  = getStatusFunction();
```

```
console.log('up')
```

>

console

callstack

Thread of *Execution*

Global *(scope)*

```
getStatusFunction:
        function {}

getStatus
function {}
```


```
const getStatus
  = getStatusFunction();

console.log('up')
```

```
> up
```

console

callstack

Thread of *Execution*

Global *(scope)*

>

console

callstack

Thread of *Execution*

Global *(scope)*

>

console