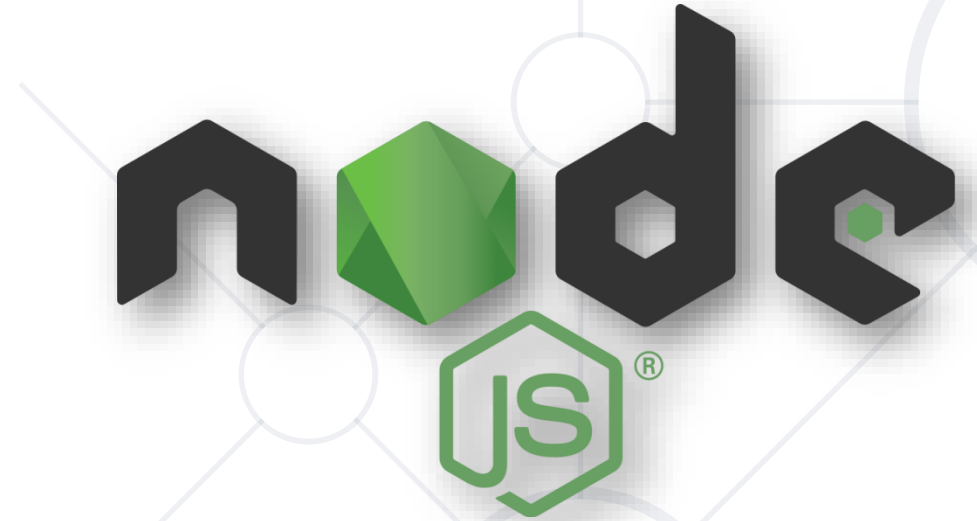
Introduction to Node.js

Overview, Modules, Web Server, Request and Response



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#js-web

Table of Contents

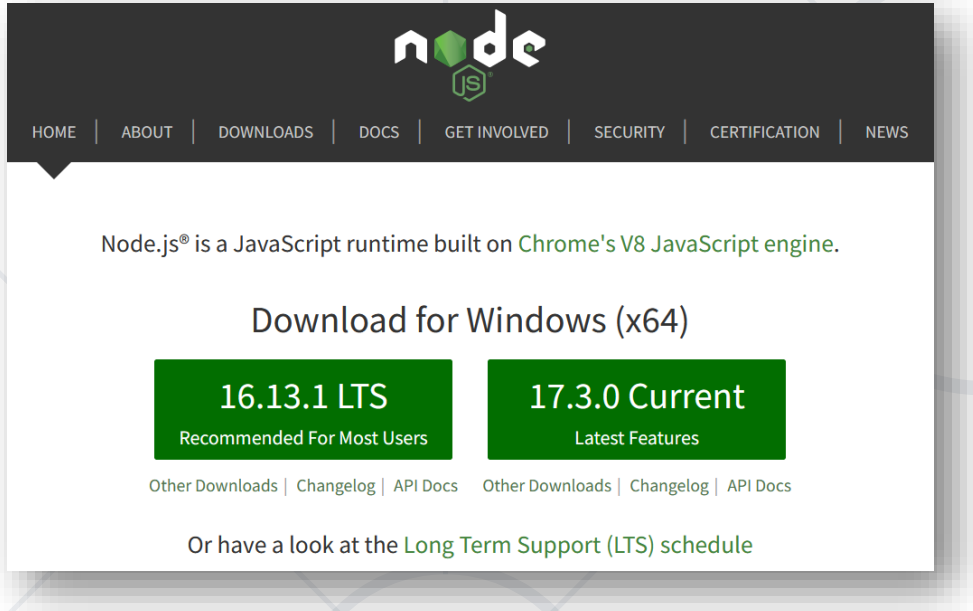
Introduction to Node.js

Node.js Overview

- A runtime environment for JS that runs on the server
- Advantages
  - **One language** for server and client
  - **Asynchronous** and **Event** Driven
  - Very **fast**
  - Efficient **package manager**

Installation

- Go to ***http://nodejs.org*** and install the latest version



- To check the currently installed version of the node, type in the **command prompt/terminal**:

```
node -v
```

Environment Setup

- From the **terminal**

```
node      // Starts REPL
let a = 5
let b = 3
a + b     // 8
```

- Interpret code from a **file**
  - Save the script to **index.js**
  - Execute from the terminal:

```
node index.js
```

NPM Packages

- Node.js **projects** are usually set up as **NPM packages**
  - From the **terminal**, inside the **target directory**

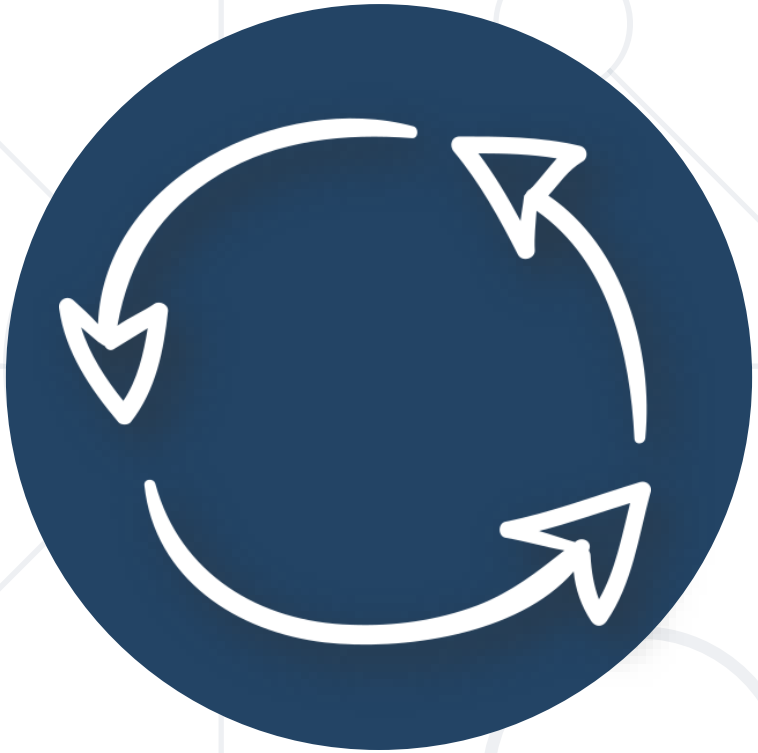
```
npm init
```

  - Answer **questions** to initialize the project
  - A **package.json** file will be created with initial configuration
  - To bypass all questions (take default values):

```
npm init -y
```

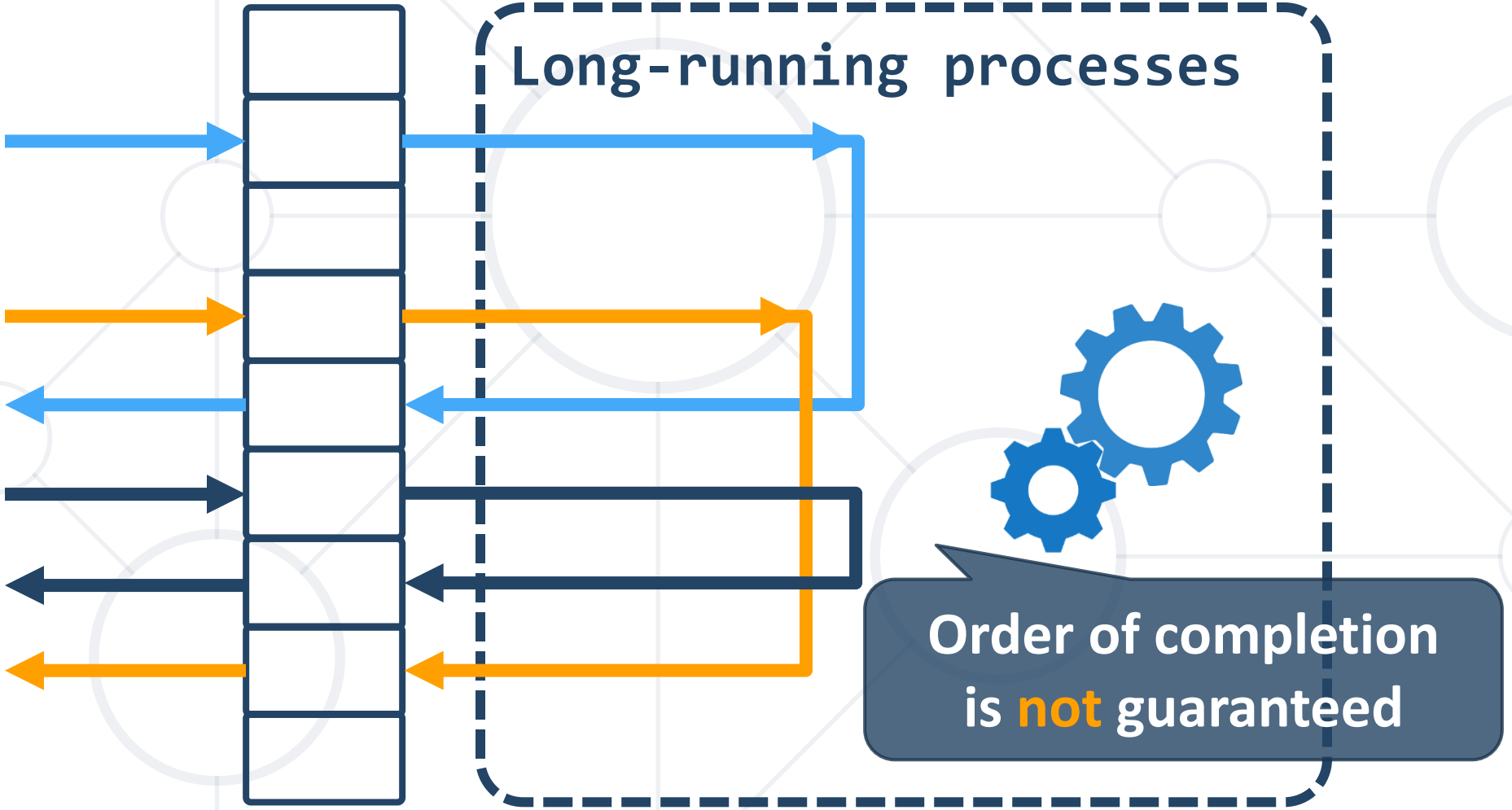
Configuration (Package.json)

```
{
  "name": "demo",
  "version": "1.0.0",
  "description": "Node.js demo project",
  "main": "index.js",
  "engines": {          // Sets versions of Node.js
    "node": ">= 6.0.0",           and other commands
    "npm": ">= 3.0.0" },
  "scripts": {          // Defines a set of node scripts
    "start": "node index.js" },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```



Event Loop

The Event Loop



Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack

Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack

bar(8)

Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack

foo(10)

bar(8)

Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack

return

bar(8)

Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack

bar(8)

Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack

return

Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack

Stack calls

Software University

```
function init(el){
    el.addEventListener(
        "click",
        handler
    );
}
```

Stack calls

Stack

Browser APIs

Hidden implementation

Stack calls

Stack

init

Browser APIs

Hidden implementation

Stack calls

Stack

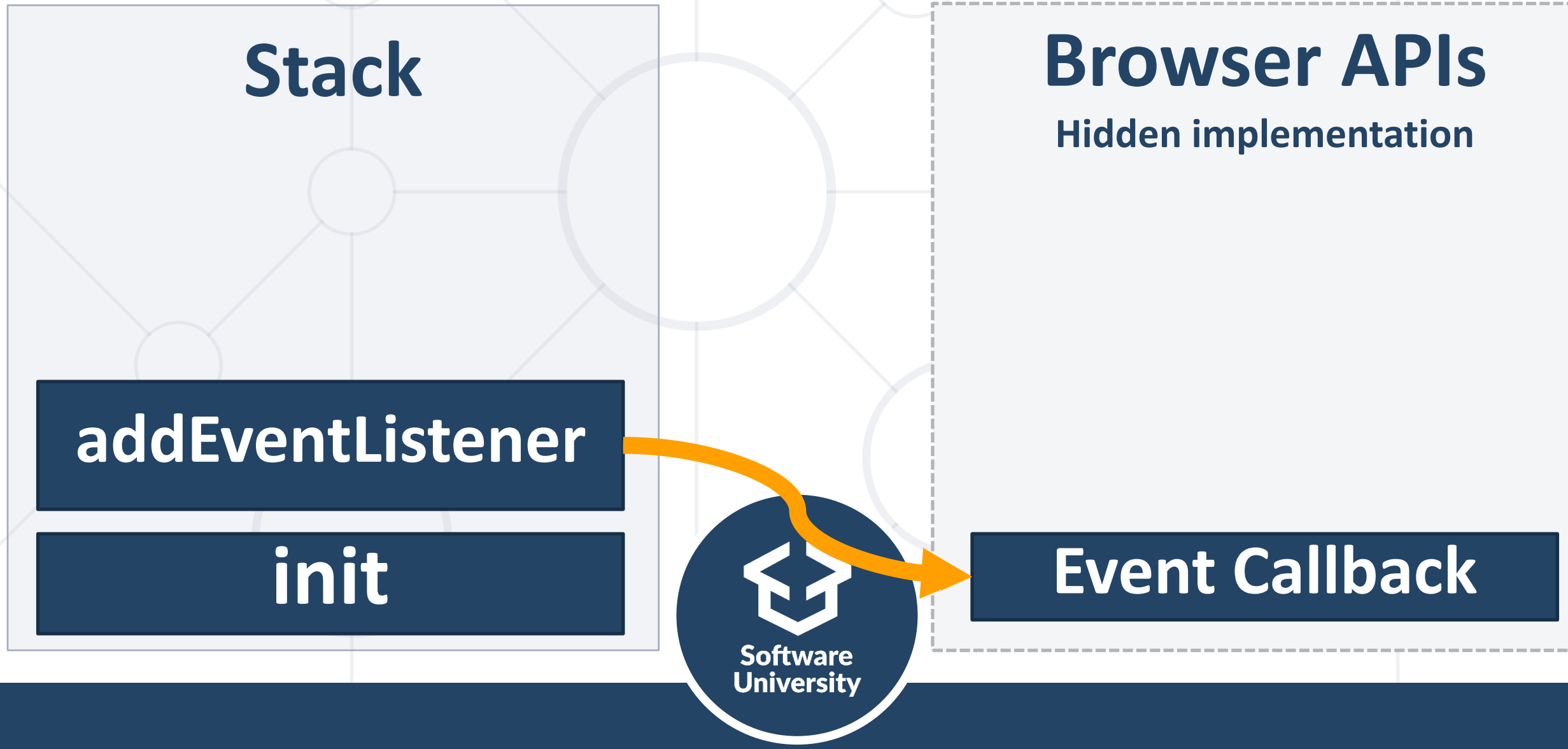
addEventListener

init

Browser APIs

Hidden implementation

Stack calls


Stack
addEventListener
init
Browser APIs
Hidden implementation
Event Callback
Software University

Stack calls

Stack

return

init

Browser APIs

Hidden implementation

Event Callback

Stack calls

Stack

return

Browser APIs

Hidden implementation

Event Callback

Stack calls

GC

Browser APIs

Hidden implementation

Event Callback

Stack calls

Stack

Browser APIs

Hidden implementation

Event Callback

Stack calls

Software University

Stack

....

Message Queue

Browser APIs

Hidden implementation

Event Callback

Stack calls

Software University

Stack

....

Event Loop

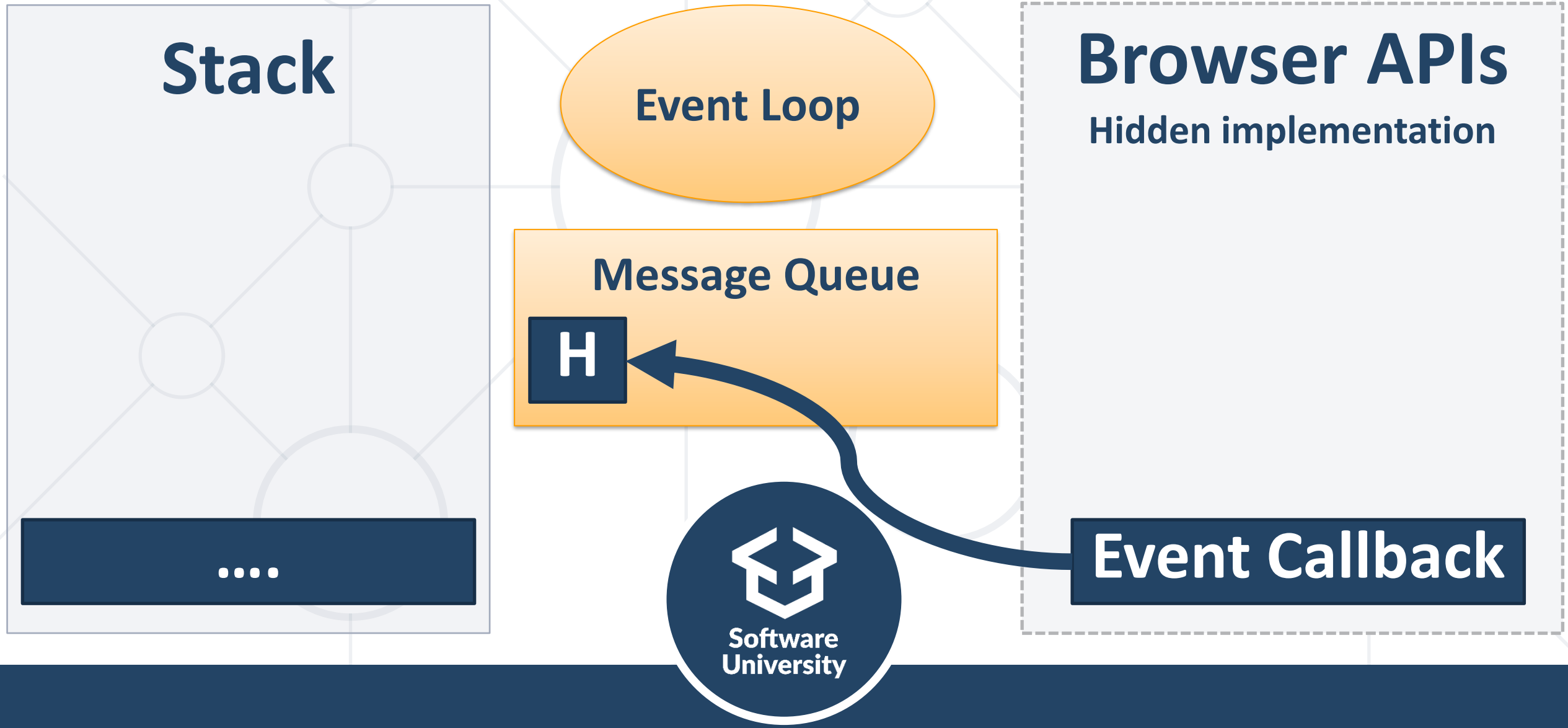
Message Queue

Browser APIs

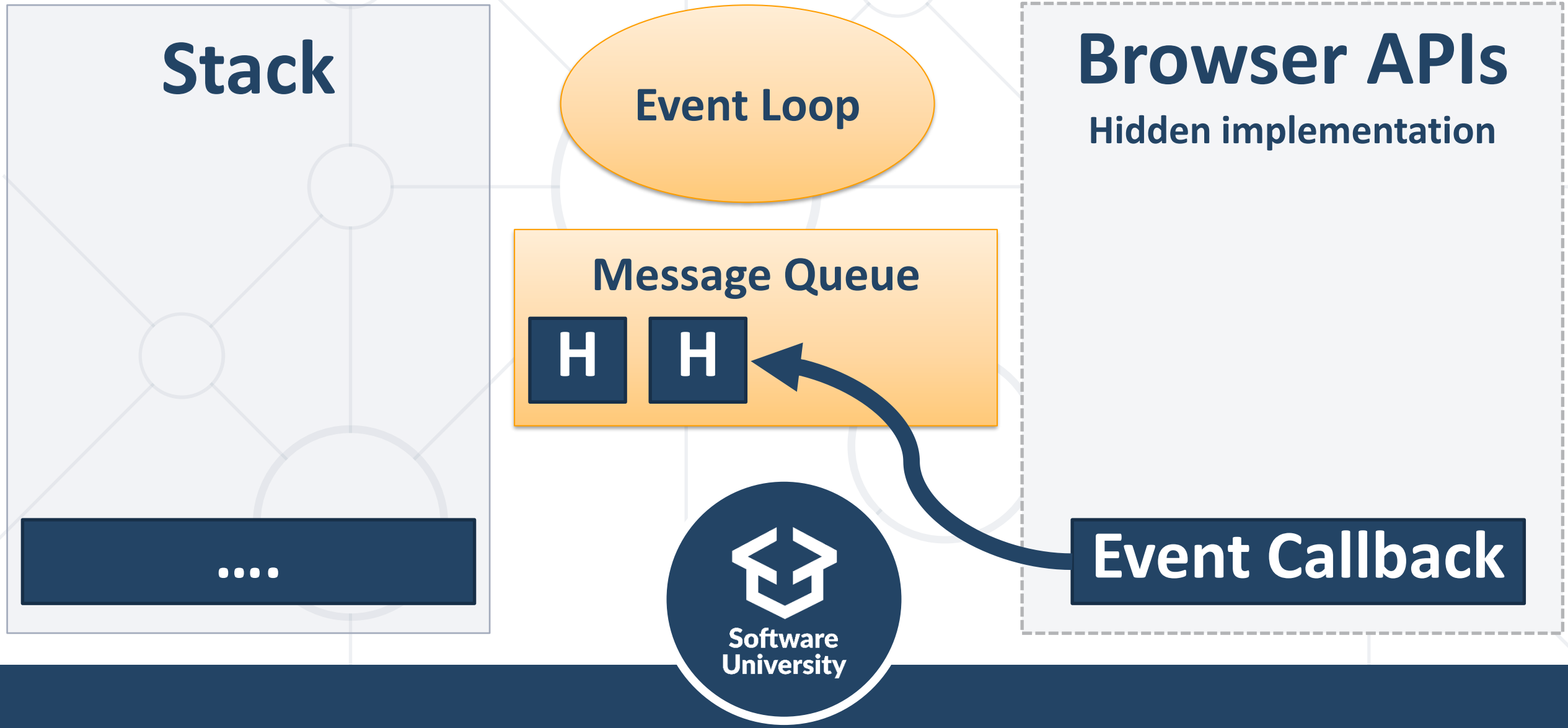
Hidden implementation

Event Callback

Stack calls


Stack
....
Event Loop
Message Queue
H
Software University
Browser APIs
Hidden implementation
Event Callback

Stack calls


Stack
....
Event Loop
Message Queue
H
H
Software University
Browser APIs
Hidden implementation
Event Callback

Stack calls

Software University

Stack

Event Loop

Message Queue

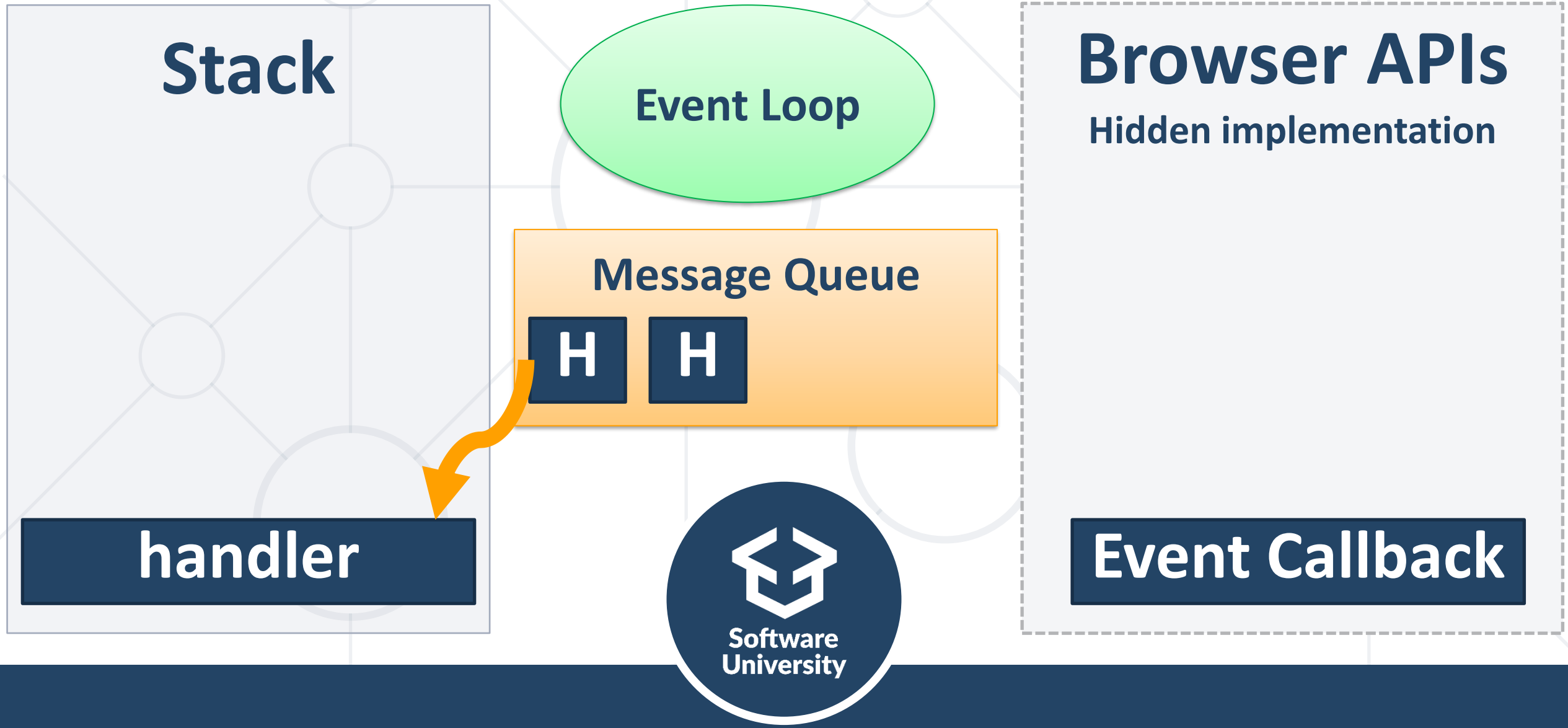
H H

Browser APIs

Hidden implementation

Event Callback

Stack calls


Stack
Event Loop
Browser APIs
Hidden implementation
Message Queue
H
H
handler
Event Callback
Software University

Stack calls

Software University

Stack

handler

Event Loop

Message Queue

H

Browser APIs

Hidden implementation

Event Callback

Stack calls

Software University

Stack

return

Event Loop

Message Queue

H

Browser APIs

Hidden implementation

Event Callback

Stack calls

Event Loop

Message Queue

H

Browser APIs

Hidden implementation

Event Callback

Stack calls

Software University

Stack

Event Loop

Message Queue

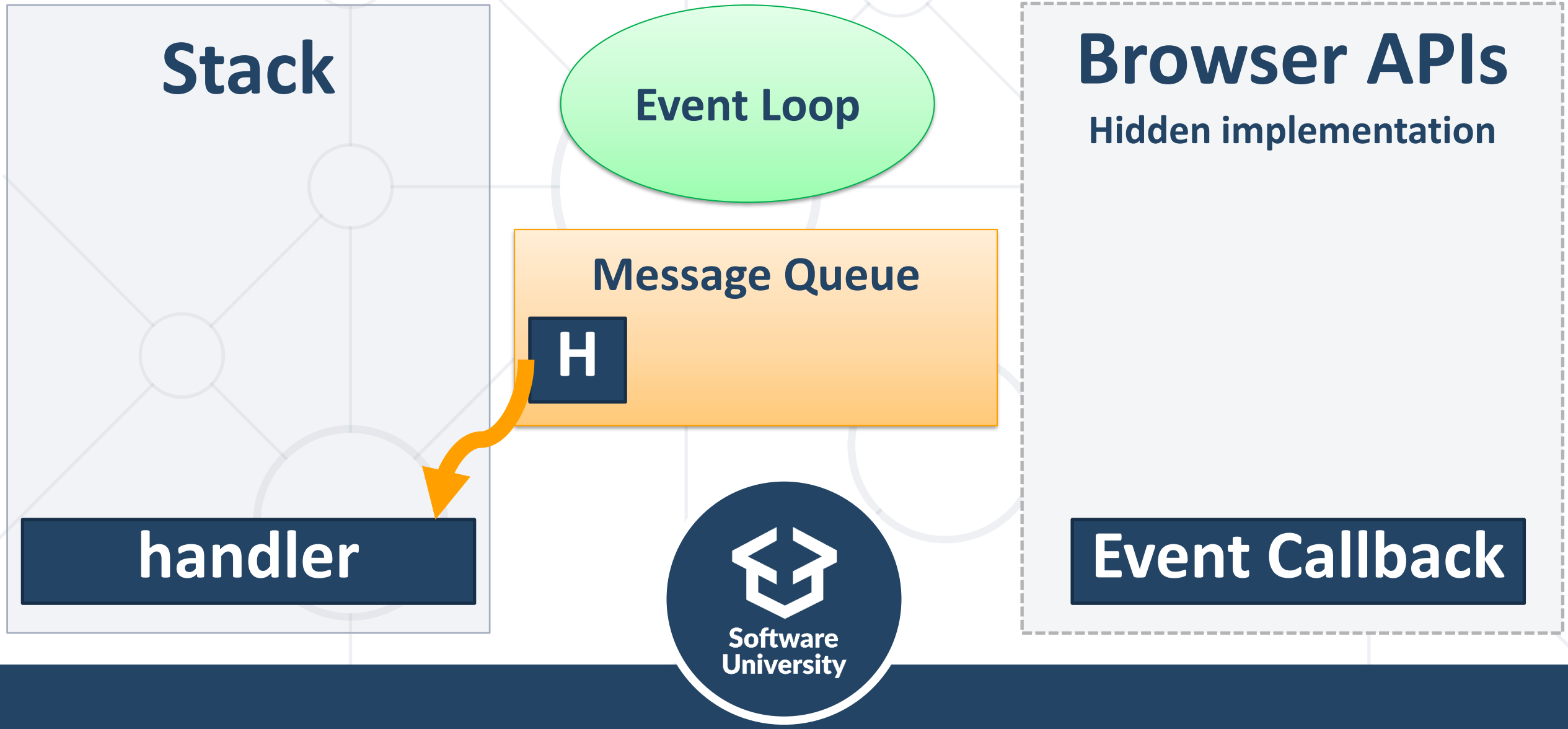
H

Browser APIs

Hidden implementation

Event Callback

Stack calls


Stack
Event Loop
Browser APIs
Hidden implementation
Message Queue
H
handler
Event Callback
Software University

Stack calls

Software University

Stack

handler

Event Loop

Message Queue

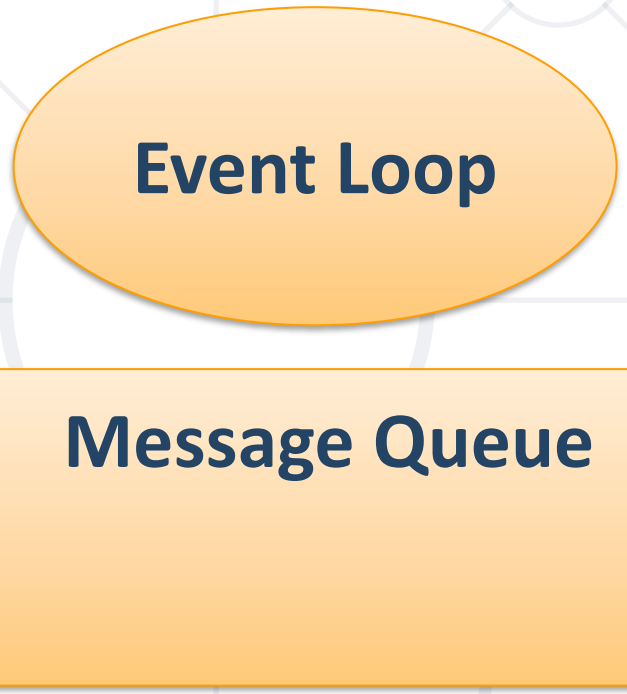
Browser APIs

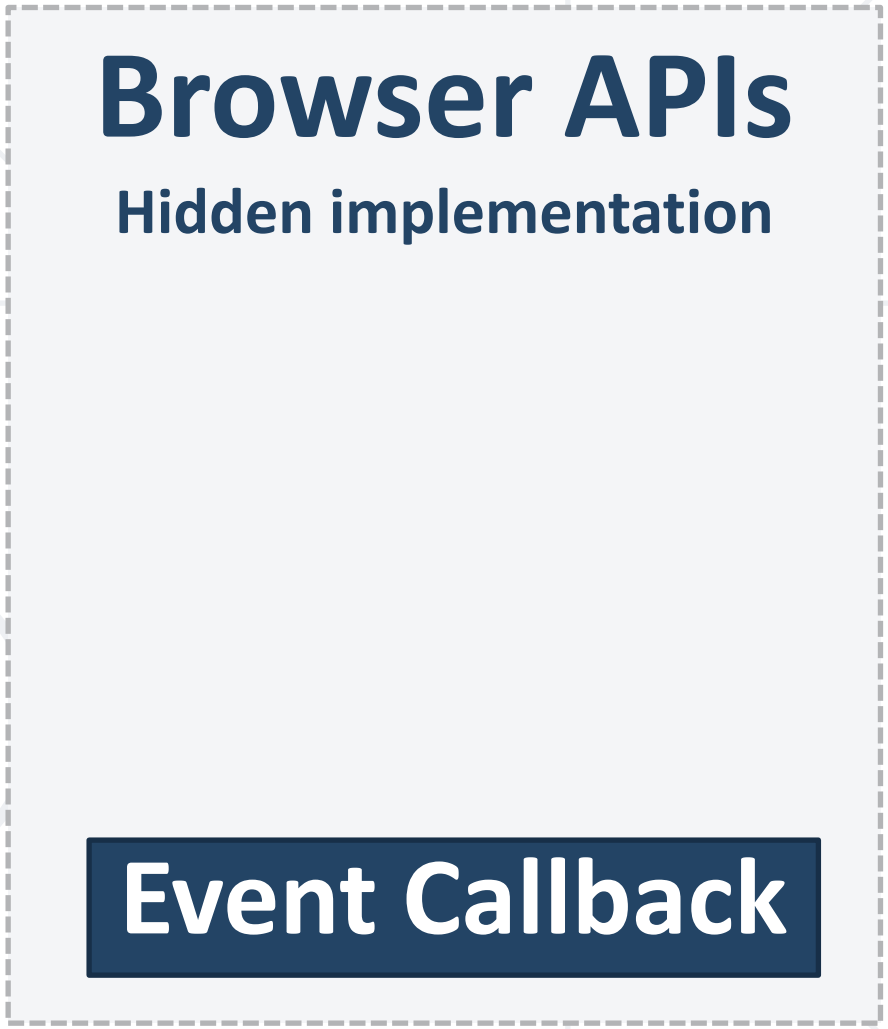
Hidden implementation

Event Callback

Stack calls


Stack
return


Event Loop
Message Queue


Browser APIs
Hidden implementation
Event Callback


Software
University

Stack calls

Event Loop

Message Queue

Browser APIs

Hidden implementation

Event Callback

Stack calls

Software University

Stack

Event Loop

Message Queue

Browser APIs

Hidden implementation

Event Callback

Modules

Modules

- Allow larger **apps** to be **split** and **organized**
- Each module has its **own context**
  - It **cannot pollute** the **global scope**
- Node.js includes **three types** of modules
  - **Core** Modules
  - **Local** Modules
  - **Third-Party** Modules

Local Modules

- Created **locally** in the Node.js application
- Include **different functionalities** in **separate** folders
- Use **`module.exports`** to expose a **function**, **object** or **variable**

```
module.exports = myModule
```

- Loaded using the **`require()`** function

```
const myModule = require('./myModule.js');
```

Third-Party Modules

- Installed from Node Package Manager (**NPM**)
- Run from the terminal

```
npm install express --save-exact
```

- To use in your code

```
const express = require('express');
```

- To install globally (for use from the terminal)

```
npm install mocha -g
```


47

Core Modules

- Includes all **functionalities** of Node.js
- Load **automatically** when Node.js process starts
- Need to be **imported** in order to be used

```
const module = require('module');
```

- Commonly used modules are
  - **http** - used to create Node.js server
  - **url**, **querystring**, **path**, **fs**

URL Module

Software University

- Provides utilities for URL **resolution** and **parsing**

```
const url = require('url');
```

- Parses an address with the **`parse()`** function
  - Returns an **object** with **info** about the **url**

```
let urlObj = url.parse(req.url);
```

- **Splits** web address into **readable** parts

URL Parts

- Host '**localhost:8080**'

```
let host = urlObj.host
```

- Path '**/home**'

```
let path = urlObj.pathname
```

- Search/query '**?year=2017&month=february**'

```
let query = urlObj.query
```

```
let search = urlObj.search
```

Query String Module

Software University

- Provides utilities for **parsing** and **formatting** URL query strings

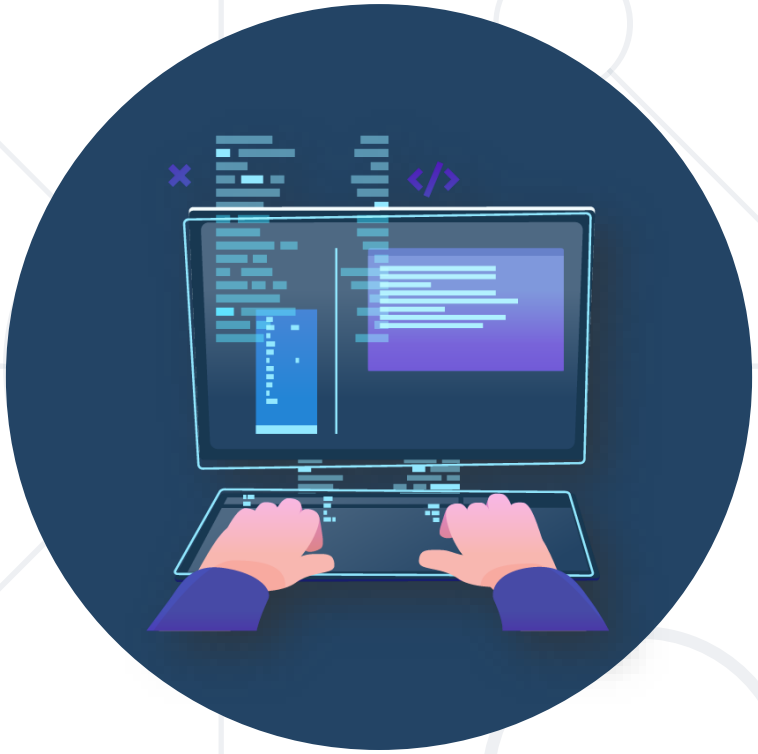
```
const queryString = require('querystring');
```

- Parses a query string into an object

```
const qs = querystring
.parse('year=2017&month=february');
```

```
const year = qs.year;          // 2017
```

```
const month = qs.month;      // february
```



Node.js Web Server

Web Servers

- All **physical** servers have **hardware**
- The hardware is controlled by the **operating system**
- **Web servers** are **software** products that use the operating system to **handle web requests**
  - Web servers **serve** Web content
- The requests are **redirected to other software** products (ASP.NET, PHP, etc.), depending on the webserver **settings**

Node.js Web Server

- Creating a simple Node.js web server

```
const http = require('http');

http.createServer((req, res) => {
  res.write('Hi!');
  res.end();
}).listen(1337);

console.log('Node.js server running on port 1337');
```

Request & Response Wrappers

The Request Wrapper

- Used to **handle** incoming http requests
- Properties
  - **httpVersion** - '1.1' or '1.0'
  - **headers** - object for request headers
  - **method** - 'GET', 'POST', etc
  - **url** - the URL of the request

Request Wrapper Example

Software University

```
const http = require('http');
const url = require('url');
const port = 1337;

http.createServer((req, res) => {
  let path = url.parse(req['url']).pathname;
  if (path === '/') {
     // TODO: Send 'Welcome to home page!'
  }
}).listen(port);
```

The Response Wrapper

- Used to **retrieve** a **response** to the **client**
- Functions
  - Create **response header**
  - Send the actual **content** to the **client**
  - **End** the response

Response Wrapper Example

```
const http = require('http');
const port = 3000;

http.createServer((req, res) => {
  res.writeHead(200, { // Response Status Code
    'Content-Type': 'text/plain'
  });
  res.write('Hello from Node.js'); // UTF-8 Encoding
  res.end(); // Always End the Response
}).listen(port);
```

Summary

- Node.js is a fast and asynchronous efficient package manager
- Applications can be organized using module
- NPM allows quick access to external modules
- Web Servers transfer resources to the Client
- The Request/Response Wrappers

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

Software University

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
  - Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg