UNIVERSITÀ DI PAVIA

Collegio Alessandro Volta
Via Adolfo Ferrata, 17, Pavia (PV)

WEB DESIGN

Lecture 6 – Node.js & Internet Laws

Giovanni Nicola D’Aloisio

Department of Engineering – University of Pavia
Department of Physics – University of Pavia

E-Mail: giovanninicola.daloisio01@universitadipavia.it

Integrating HTML5 & JS

- With the introduction of HTML5, JS got some interesting features.
- The system notifications request user's attention, showing a short message, sometimes outside the current page, even if we are not directly using the web browser.
- Here on the right there's a sample code for asking permission to show notifications and for showing one.
- Modern web browser stop the notification if there is not the first part of this code (asking for permission to the user), so don't miss it.

HTML + JS (speech recognition)

```
<input type="text" x-webkit-speech />

<script>
var recognition = new SpeechRecognition();
var speechRecognitionList = new SpeechGrammarList();
</script>
```

HTML + JS (system notifications)

```
<button id="btn-asking">Richiedi permessi</button>
<button id="btn-notify">Mostra notifica</button>

<script>
// the browser searches labelled buttons inside HTML
var btnAsking = document.getElementById( 'btn-asking' );
var btnNotify = document.getElementById( 'btn-notify' );

// after the click, the browser asks for permission
btnAsking.addEventListener( 'click', function() {
    Notification.requestPermission();
} );

// after the click, the browser shows the notification
btnNotify.addEventListener( 'click', function() {
    new Notification( “New notification!" );
} );
</script>
```

Integrating HTML5 with JS – Front-end examples

HTML + JS (drag and drop)

```
// Drag and Drop of files from a web page to the device

<a href="src/star.mp3" draggable="true" class="dragout"
   data-downloadurl="MIMETYPE:FILENAME:ABSOLUTE_URI_TO_FILE">download</a>

<script>
var files = document.querySelectorAll('.dragout');
for (var i = 0, file; file = files[i]; ++i) {
  file.addEventListener('dragstart', function(e) {
    e.dataTransfer.setData('DownloadURL', this.dataset.downloadurl);
  }, false);
}
</script>
```

JavaScript (geopositioning)

```
// For declaring your position to your web application

if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(function(posi
tion) {
    var latLng = new google.maps.LatLng(
        position.coords.latitude,
position.coords.longitude);
    var marker = new google.maps.Marker({position:
latLng, map: map});
    map.setCenter(latLng);
  }, errorHandler);
}
```

JavaScript (file system API)

```
// Asynchronously write a file inside a file system in sandbox mode

window.requestFileSystem(window.TEMPORARY, 1024 * 1024, function(fs) {
  fs.root.getFile('log.txt', {create: true}, function(fileEntry) {
    fileEntry.createWriter(function(writer) { .
        writer.onwrite = function(e) { ... };
        writer.onerror = function(e) { ... };
        var bb = new BlobBuilder();
        bb.append('Hello World!');
        writer.write(bb.getBlob('text/plain'));
    }, opt_errorHandler);
  }
}, opt_errorHandler);
```

JavaScript (WebGL)

```
// For showing a 3D model using WebGL (or Web 3D)

function main() {
  const canvas = document.querySelector("#glCanvas");
  const gl = canvas.getContext("webgl");
  if (gl === null) {
    alert("Aggiorna il tuo browser");
    return;
  }
  gl.clearColor(0.0, 0.0, 0.0, 1.0);
  gl.clear(gl.COLOR_BUFFER_BIT);
}
window.onload = main;
```

Integrating HTML5 with JS – Back-end examples

JavaScript (web socket)

```
// Bi-directional connection full-duplex on the Web

var socket = new WebSocket('www.sito.it');
socket.onopen = function(event) {
  socket.send('Ciao');
};
socket.onmessage = function(event) { alert(event.data); }
socket.onclose = function(event) { alert('chiuso'); }
```

JavaScript (Local Storage & other)

```
// For navigating between downloaded pages in offline mode

saveButton.addEventListener('click', function () {
  window.localStorage.setItem('value', area.value);
  window.localStorage.setItem('timestamp', (new
Date()).getTime());
}, false);
textarea.value = window.localStorage.getItem('value’);

<html manifest="cache.appcache">
window.applicationCache.addEventListener('updateready',
function(e) {
  if (window.applicationCache.status ==
window.applicationCache.UPDATEREADY) {
    window.applicationCache.swapCache();
    if (confirm('A new version of this site is available. Load
it?')) {
      window.location.reload();
    }
  }
}, false);
```

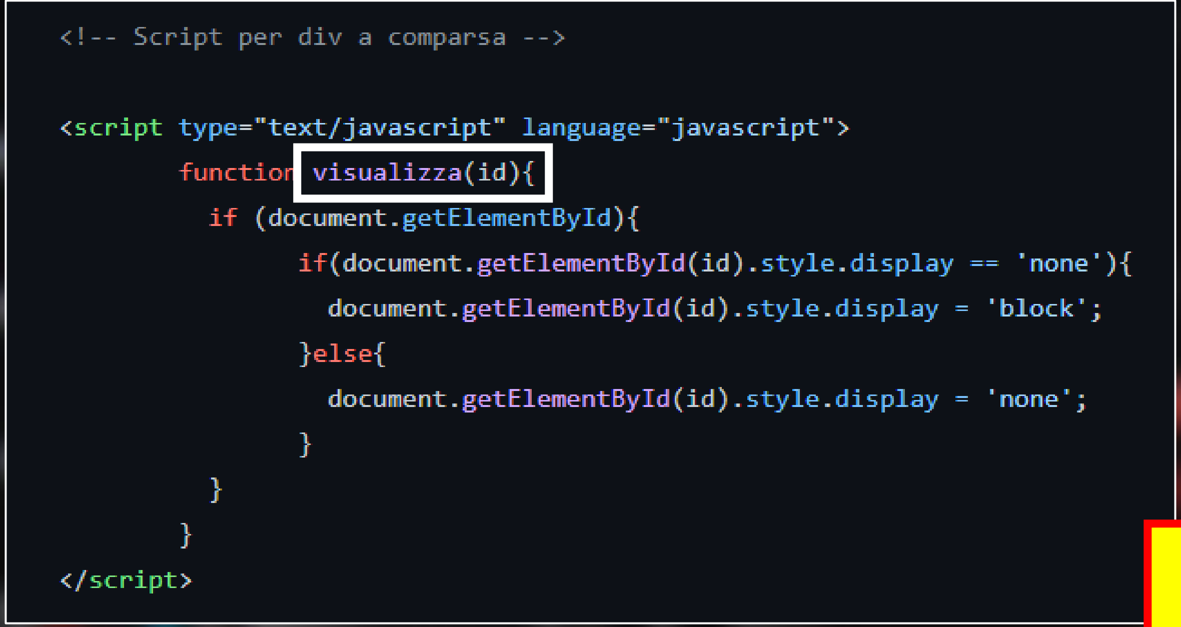
JavaScript (web workers)

```
// They increment web site’s performance

var worker = new Worker('task.js');
worker.onmessage = function(event) { alert(event.data); };
worker.postMessage('data');
task.js:
self.onmessage = function(event) {
  // Do some work.
  self.postMessage("recv'd: " + event.data);
};
```

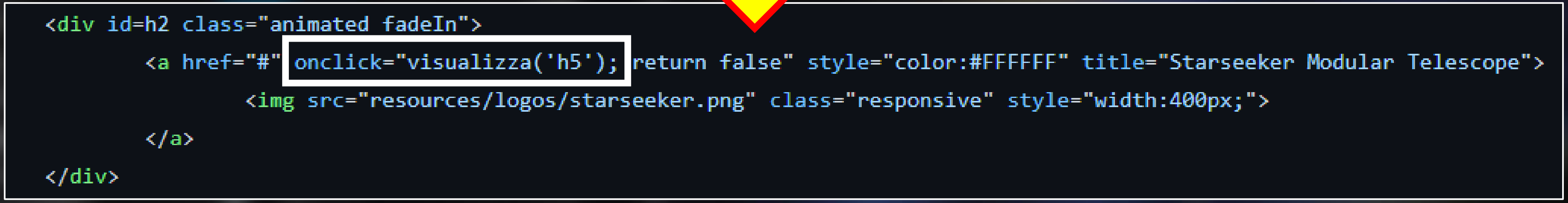
JavaScript (IndexedDB)

```
// Using the DB indexes, it enables high efficiency data queries.
// Although storing web data is useful only with less data,
// IndexedDB extends its potentiality.

window.requestFileSystem(window.TEMPORARY, 1024 * 1024, function(fs) {
  fs.root.getFile('log.txt', {create: true}, function(fileEntry) {
    fileEntry.createWriter(function(writer) { .
        writer.onwrite = function(e) { ... };
        writer.onerror = function(e) { ... };
        var bb = new BlobBuilder();
        bb.append('Hello World!');
        writer.write(bb.getBlob('text/plain'));
    }, opt_errorHandler);
  }
}, opt_errorHandler);
```

Returning to simple examples…

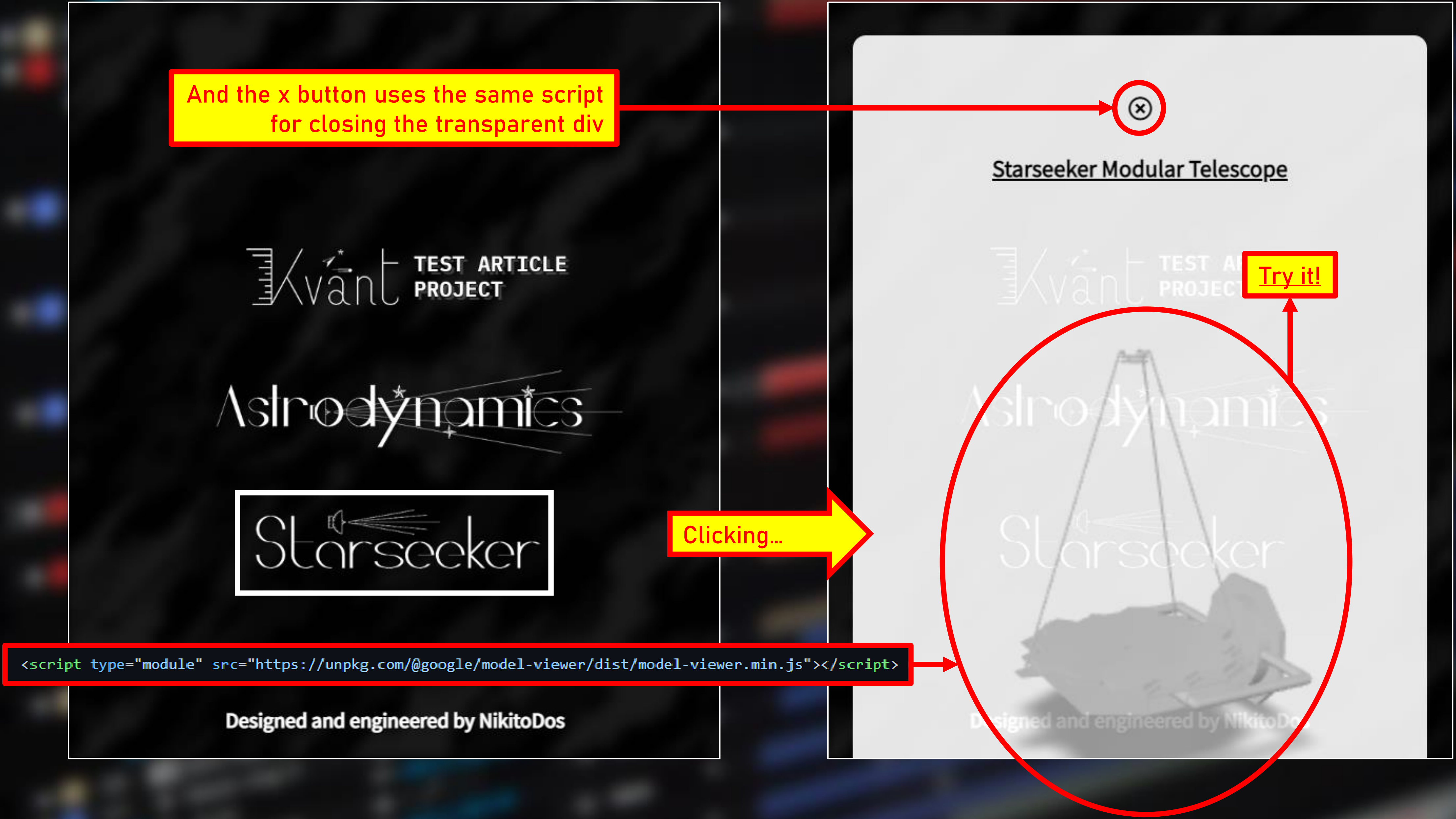


```
<!-- Script per div a comparsa -->

<script type="text/javascript" language="javascript">
        function visualizza(id){
          if (document.getElementById){
                if(document.getElementById(id).style.display == 'none'){
                  document.getElementById(id).style.display = 'block';
                }else{
                  document.getElementById(id).style.display = 'none';
                }
          }
        }
</script>
```



```
<div id=h2 class="animated fadeIn">
        <a href="#" onclick="visualizza('h5'); return false" style="color:#FFFFFF" title="Starseeker Modular Telescope">
                <img src="resources/logos/starseeker.png" class="responsive" style="width:400px;">
        </a>
</div>
```

…see the result



And the x button uses the same script for closing the transparent div

Kvant TEST ARTICLE PROJECT

Astrodynamics

Starseeker

Clicking...

```
<script type="module" src="https://unpkg.com/@google/model-viewer/dist/model-viewer.min.js"></script>
```

Designed and engineered by NikitoDos

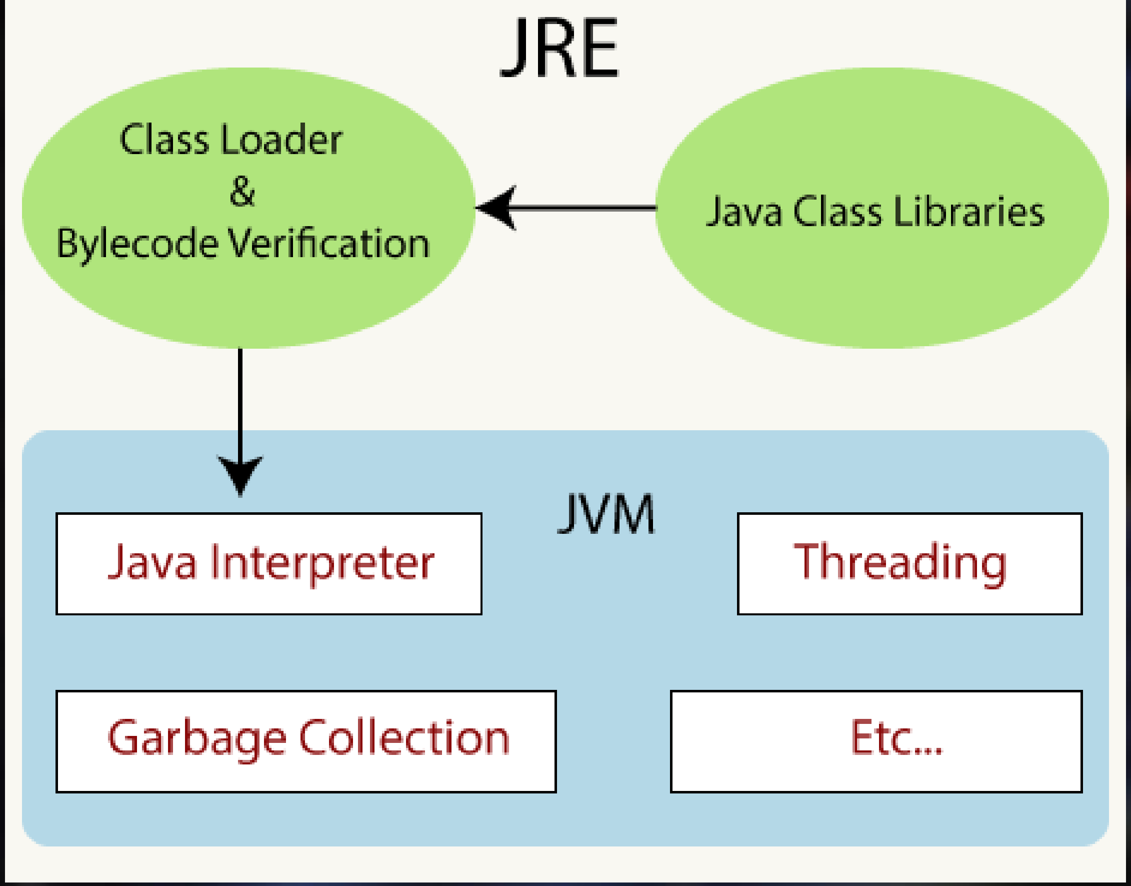
Starseeker Modular Telescope

Try it!

Run-time systems

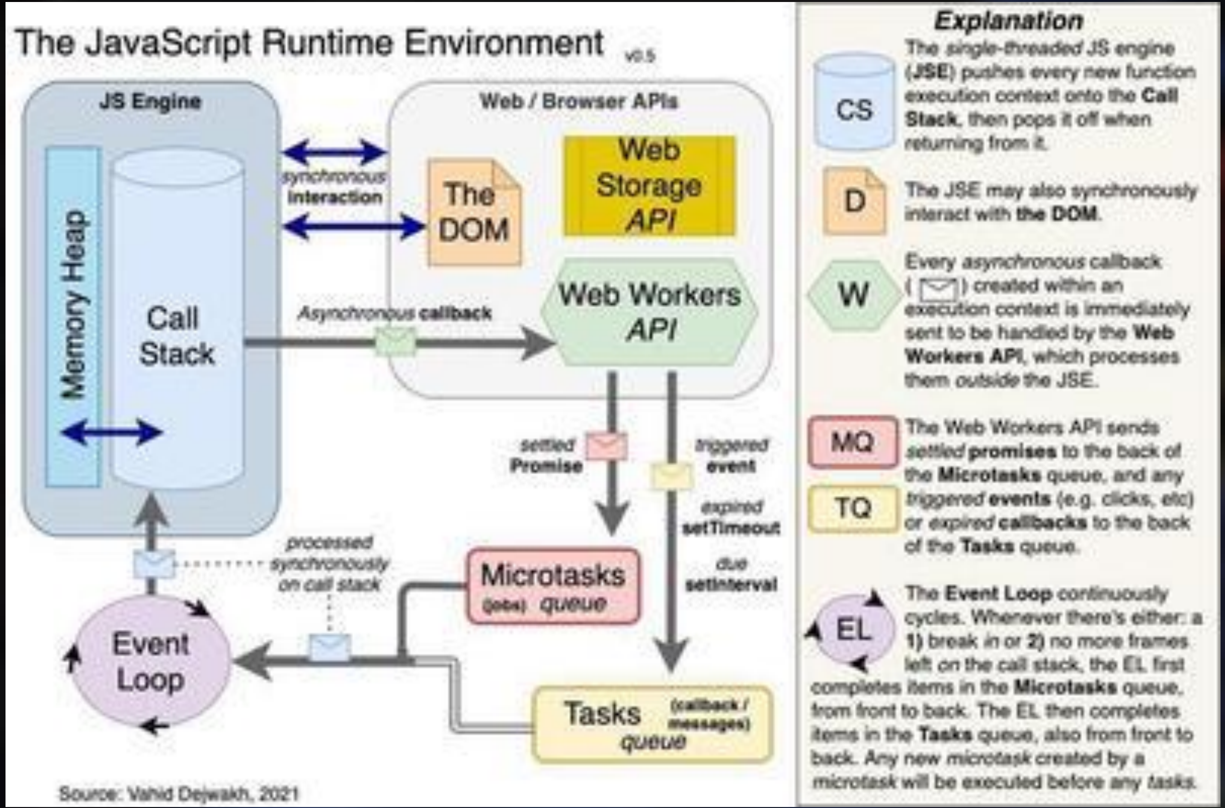
- A runtime system or runtime environment is a sub-system that exists both in the computer where a program is created, as well as in the computers where the program is intended to be run.

- The name comes from the compile time and runtime division from compiled languages, which similarly distinguishes the computer processes involved in the creation of a program (compilation) and its execution in the target machine (the run time).



JavaScript engines

- To start writing JavaScript we don't need to install any specific software because each web browser has its own version of the JS engine that parses the code for us.
- Chrome uses the V8 JS engine which has been developed by the Chromium Project.
- The JS engine translates source code that developers write into machine code that allows a computer to perform specific tasks.



The JavaScript Runtime Environment v0.5

Explanation

The *single-threaded* JS engine (**JSE**) pushes every new function execution context onto the **Call Stack**, then pops it off when returning from it.

The JSE may also synchronously interact with **the DOM**.

Every *asynchronous* callback (✉) created within an execution context is immediately sent to be handled by the **Web Workers API**, which processes them *outside* the JSE.

The Web Workers API sends *settled* **promises** to the back of the **Microtasks** queue, and any *triggered* **events** (e.g. clicks, etc) or *expired* **callbacks** to the back of the **Tasks** queue.

The **Event Loop** continuously cycles. Whenever there's either: a **1)** break *in* or **2)** no more frames left on the call stack, the EL first completes items in the **Microtasks** queue, from front to back. The EL then completes items in the **Tasks** queue, also from front to back. Any new *microtask* created by a microtask will be executed before any *tasks*.

Source: Vahid Dejwakh, 2021

The Chromium V8 engine

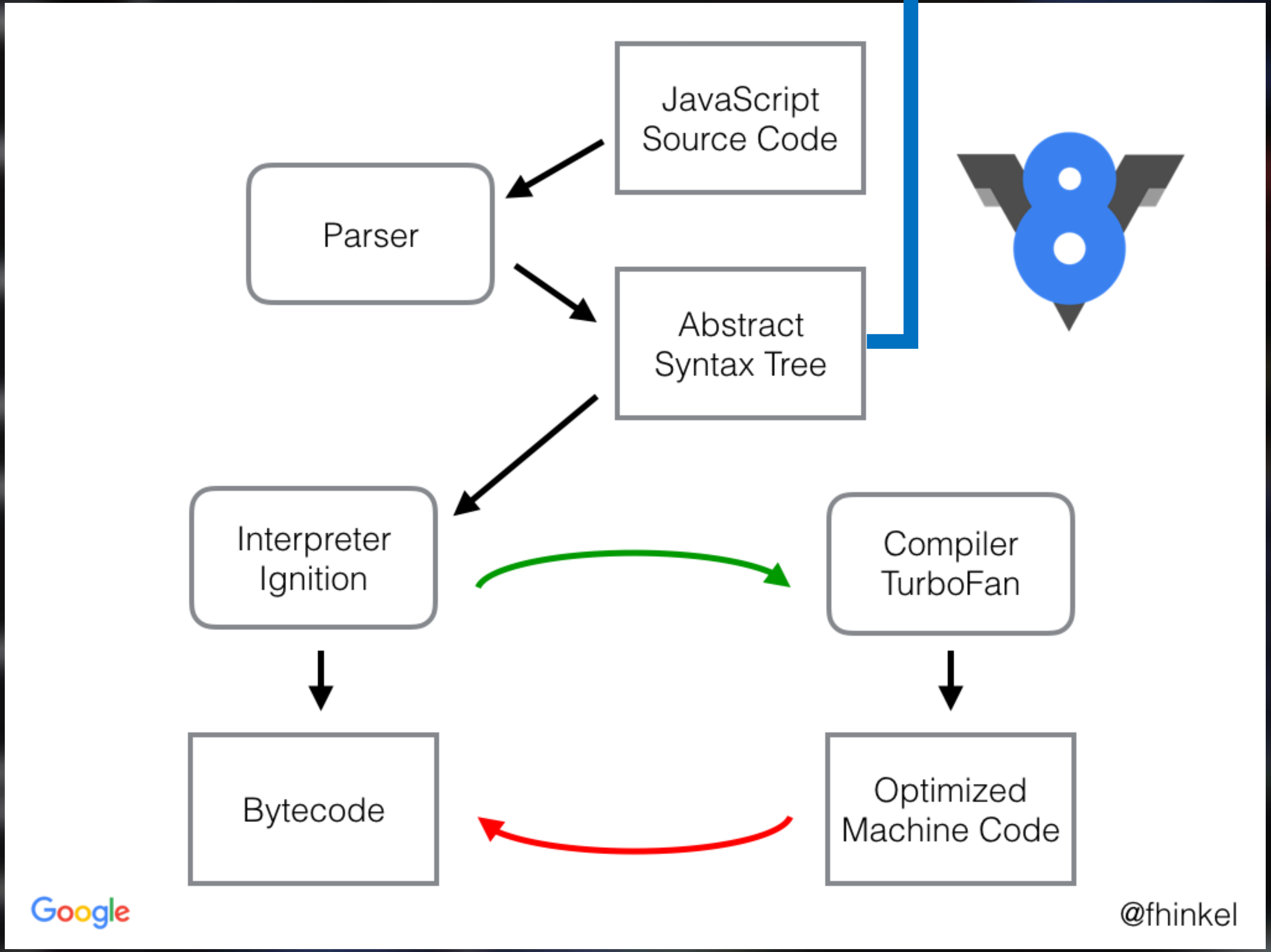
- V8 is written in C++. It is an open source project by Google, and it can be used on the client and server side.

- It is like all other JavaScript engines, but tuned for high performance.

- Recent benchmarks have put its speed ahead of PHP, Ruby and Python, and potentially higher than C.

- The *parser* generates an abstract syntax tree. *Ignition* generates bytecode from this syntax tree. Finally, *TurboFan* eventually takes the bytecode and generates optimized machine code from it.

A tree representation of the syntactic structure of the JS code.



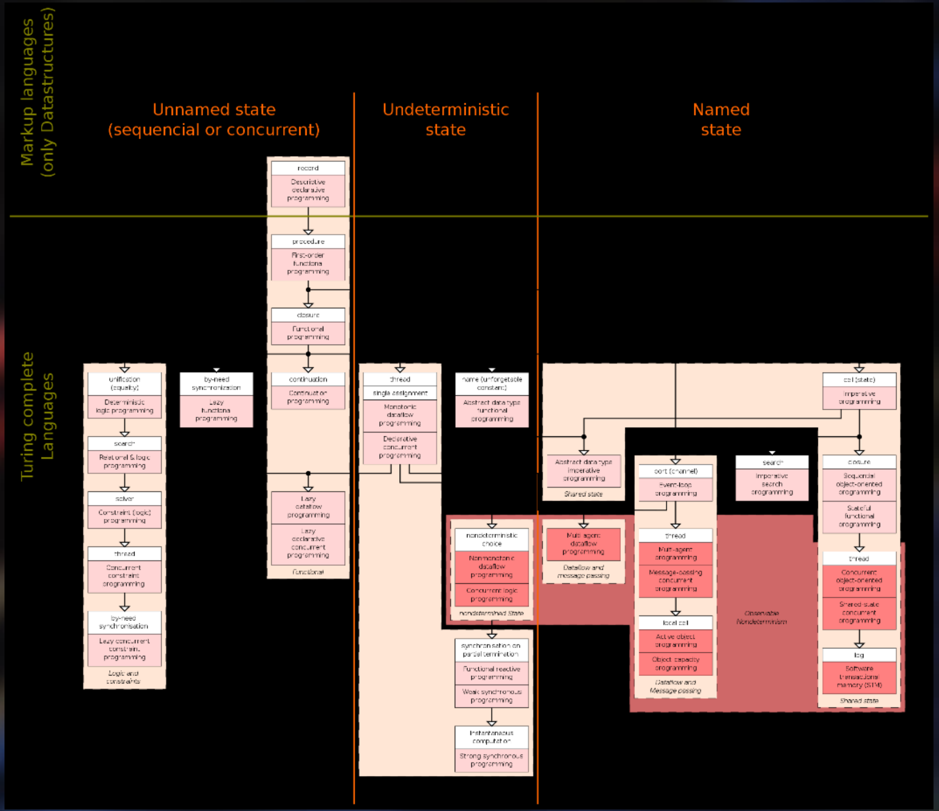
Programming paradigms

- Programming paradigms are a way to classify programming languages based on their features.

- Some paradigms are concerned mainly with implications for the execution model of the language; others with the way that code is organized. Yet others are concerned mainly with the style of syntax.

- imperative programming - the programmer instructs the machine how to change its state;

- declarative programming - the programmer merely declares properties of the desired result.

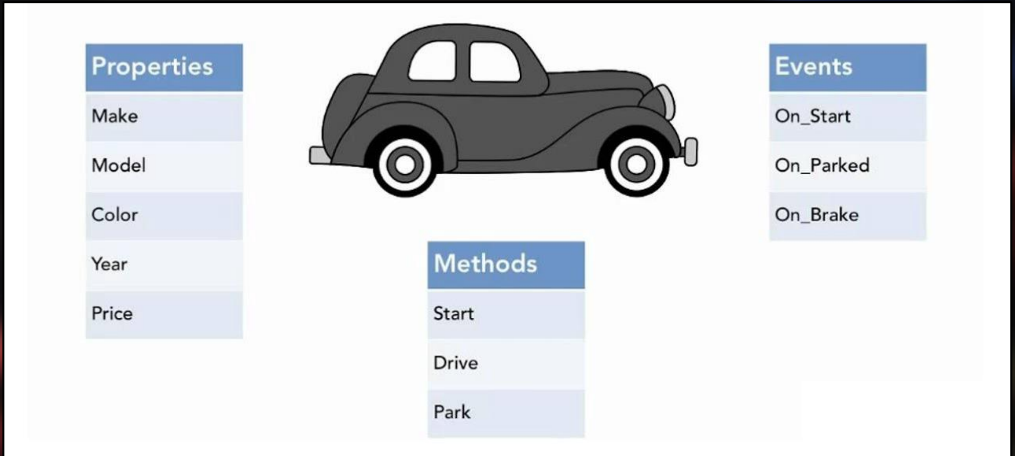


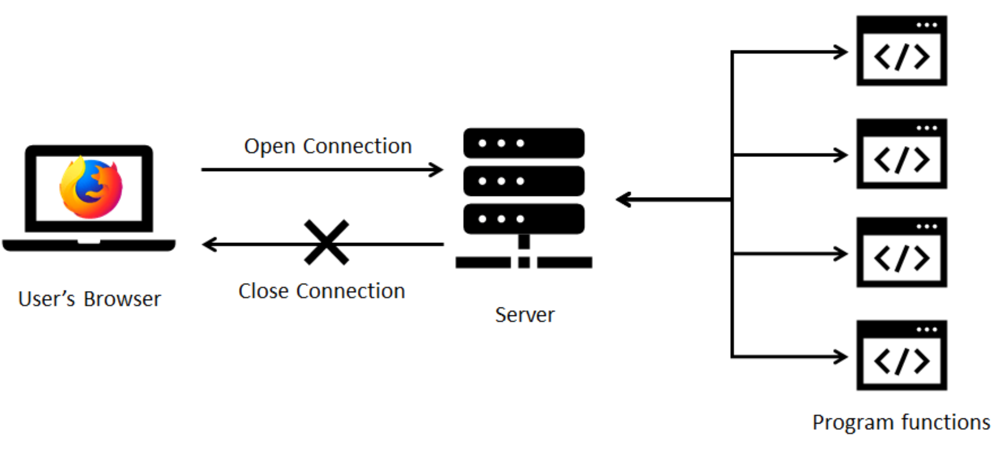
Markup languages (only Datastructures)
Turing complete Languages
Unnamed state (sequencial or concurrent)
Undeterministic state
Named state
record
Descriptive declarative programming
procedure
First-order functional programming
closure
Functional programming
continuation
Continuation programming
Lazy dataflow programming
Lazy declarative concurrent programming
Functional
unification (equality)
Deterministic logic programming
search
Relational & logic programming
solver
Constraint (logic) programming
thread
Concurrent constraint programming
by-need synchronisation
Lazy concurrent constraint programming
Logic and constraints
by-need synchronization
Lazy functional programming
thread
single assignment
Monotonic dataflow programming
Declarative concurrent programming
name (unforgetable constant)
Abstract data type functional programming
nondeterministic choice
Nonmonotonic dataflow programming
Concurrent logic programming
nondetermined State
synchronisation on partial termination
Functional reactive programming
Weak synchronous programming
Instantaneous computation
Strong synchronous programming
Abstract data type imperative programming
Shared state
Multi agent dataflow programming
Dataflow and message passing
port (channel)
Event-loop programming
thread
Multi-agent programming
Message-passing concurrent programming
local cell
Active object programming
Object capacity programming
Dataflow and Message passing
search
Imperative search programming
Observable Nondeterminism
cell (state)
Imperative programming
closure
Sequential object-oriented programming
Stateful functional programming
thread
Concurrent object-oriented programming
Shared-state concurrent programming
log
Software transactional memory (STM)
Shared state

Object-oriented vs event-oriented programming

- In OOP the procedures (or methods) are attached to the objects and can access and modify the object's data fields. OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

- In event-driven programming the flow of the program is determined by events such as user actions, sensor outputs, or message passing from other programs or threads. Event-driven programming is the dominant paradigm used in graphical user interfaces and other applications that are centered on performing certain actions in response to user input.



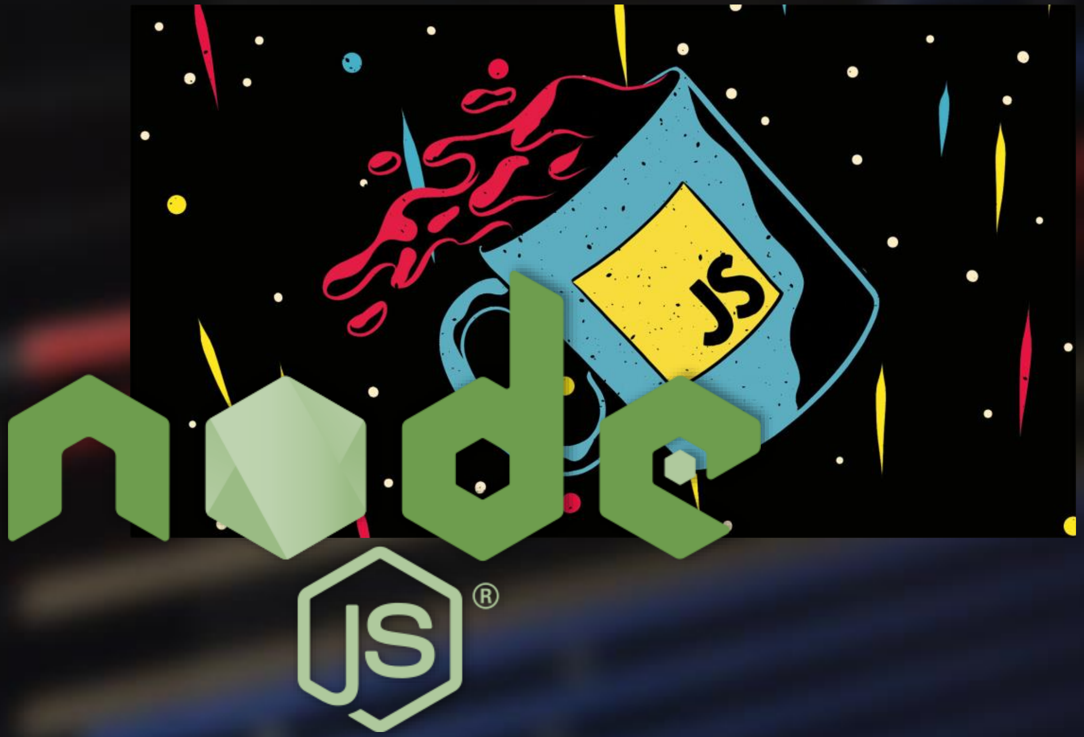


What is Node.js?

- Node.js is an open source multi-platform event-driven runtime system for executing JavaScript code, built on the V8 engine.

- Many of its basic modules are JS-coded, and the developers can write new modules in the same language.

- Some of the enterprises that support the program are GoDaddy, Groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Rakuten, SAP, Voxer, Walmart e Yahoo!.

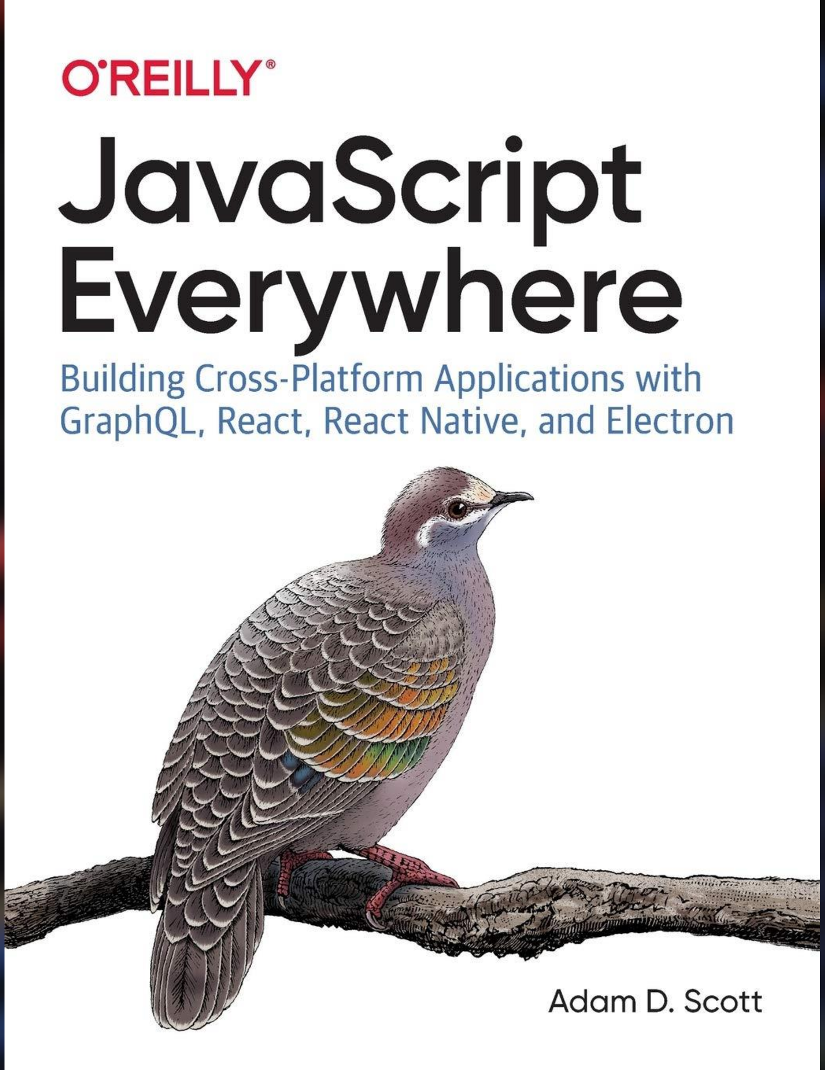


Why we study Node.js

- Node.js allows to use JS for writing server-side programs, for example for producing the content of dynamic web pages, before the page itself is sent to the user.

- Thanks to this, Node.js allows to incorporate the "JavaScript everywhere" paradigm, unifying the Web app development around a single programming language (JS).

- Node.js is event-driven, so it allows the asynchronous I/O (input-output), optimizing throughput and scalability in web applications which require lots of I/O operations, like real-time games.



O'REILLY

JavaScript Everywhere

Building Cross-Platform Applications with GraphQL, React, React Native, and Electron

Adam D. Scott

First Node.js example with HTTP Module

- Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).

- The require() method includes the HTTP module. Then, HTTP module can create an HTTP server that listens to server ports and gives a response back to the client, with the createServer() method.

- The function passed into the http.createServer() method, will be executed when someone tries to access the computer on port 3000.

- You can see an online demo of this example here, or you can try it by yourself offline.

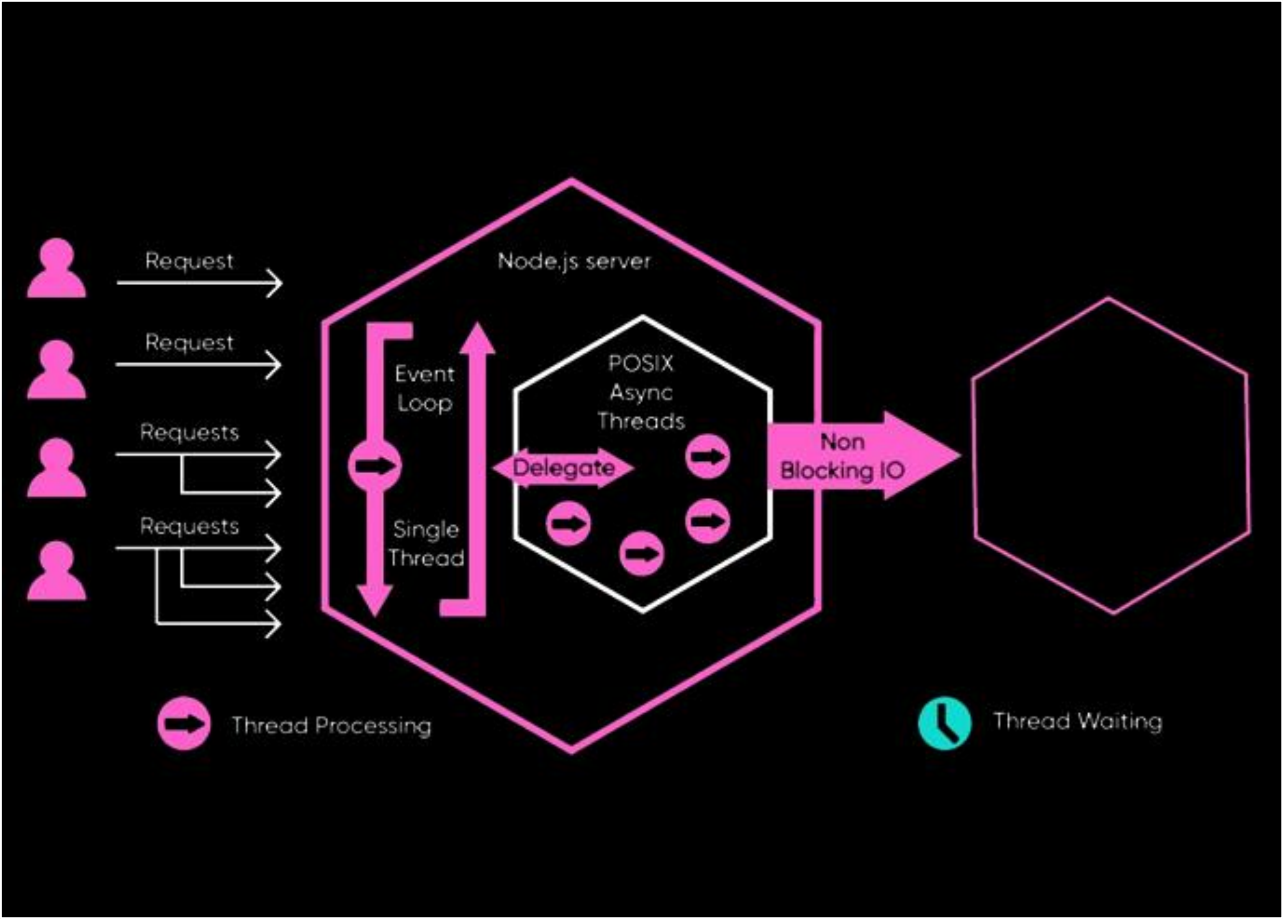
demo_http.js

```
const http = require('http’); // I create a fixed var

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at
http://${hostname}:${port}/`);
});
```

In this example many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.

How Node.js works

- It is in contrast to today's more common concurrency model, in which OS threads are employed.
- Thread-based networking is inefficient and difficult to use.
- Node.js doesn't directly perform I/O, so the process never blocks except when the I/O is performed using synchronous methods of standard libraries.
- This is why scalable systems are very reasonable to develop in Node.js.



Node.js Upload Files and File System modules

There is a very good module for working with file uploads, called "Formidable“, which can be downloaded and installed using NPM.

- Step 1: Create an Upload Form - Create a Node.js file that writes an HTML form, with an upload field.
- Step 2: Parse the uploaded file - The Formidable module enable parsing of uploaded files once they reach the server. When the file is parsed, it gets placed on a temporary folder on your computer.
- Step 3: Save the File - To move the file to the folder of your choice, use the File System module, and rename the file.

upload_files.js

```
var http = require('http');
var formidable = require('formidable');
var fs = require('fs');

http.createServer(function (req, res) {
  if (req.url == '/fileupload') {
    var form = new formidable.IncomingForm();
    form.parse(req, function (err, fields, files) {
      var oldpath = files.filetoupload.filepath;
      var newpath = 'C:/Users/Your Name/' +
files.filetoupload.originalFilename;
      fs.rename(oldpath, newpath, function (err) {
        if (err) throw err;
        res.write('File uploaded and moved!');
        res.end();
      });
 });
  } else {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<form action="fileupload" method="post"
enctype="multipart/form-data">');
    res.write('<input type="file" name="filetoupload"><br>');
    res.write('<input type="submit">');
    res.write('</form>');
    return res.end();
  }
}).listen(8080);
```

Further information

There's a lot of documentation on the web; anyway, I advice you the following:

- W3C Schools;
- MDN Web Docs;
- Official docs;
- HTML.it.

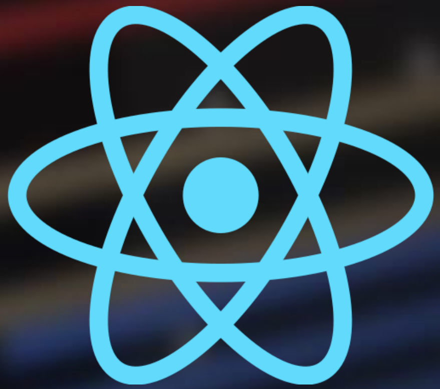
On the right: usage of nodemailer module, for easily sending emails from your server.
Use the username and password from your selected email provider to send an email.

send_email.js

```
var nodemailer = require('nodemailer');

var transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: 'youremail@gmail.com',
    pass: 'yourpassword'
  }
});

var mailOptions = {
  from: 'youremail@gmail.com',
  to: 'myfriend@yahoo.com',
  subject: 'Sending Email using Node.js',
  text: 'That was easy!'
};

transporter.sendMail(mailOptions, function(error, info){
  if (error) {
    console.log(error);
  } else {
    console.log('Email sent: ' + info.response);
  }
});
```

Not only Node.js...

Click on the symbols for more information.



Good and bad news

The good one:

We will not code anymore, from this point.

The bad one:

We're about to talk about security and laws.

Since the Internet is relatively new and evolving, legal systems can struggle to keep up, and often include and apply principles from different legal fields, which may be ambiguous.

I can calculate the motions of the heavenly bodies, but not the madness of the people.

Sir. Isaac Newton (disputed)

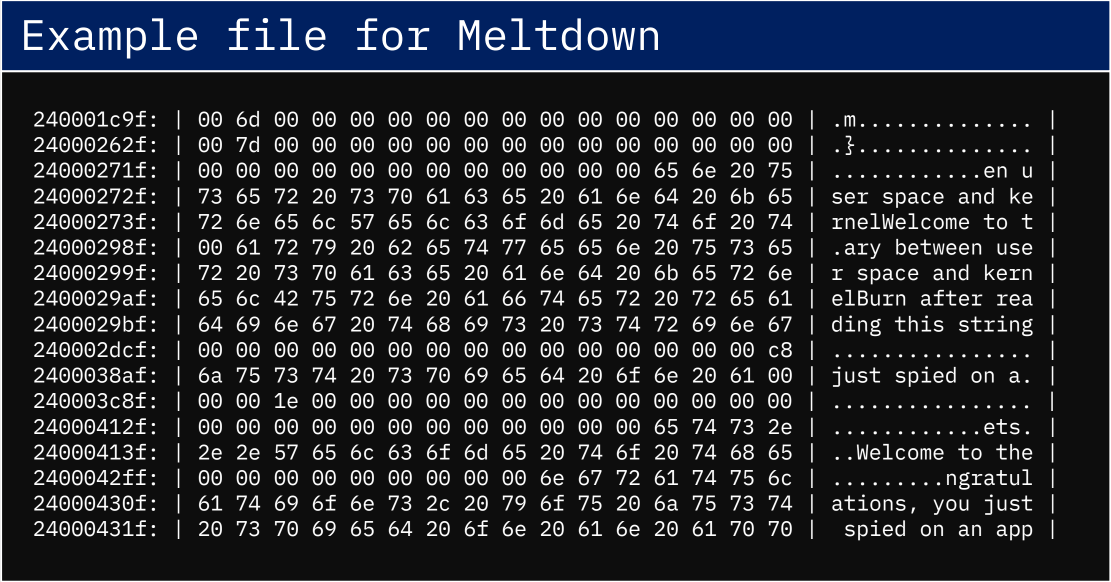
A recent security case: Meltdown & Spectre

Meltdown breaks the most fundamental isolation between user applications and the operating system. This attack allows a program to access the memory, and thus also the secrets, of other programs and the operating system.

Spectre breaks the isolation between different applications. It allows an attacker to trick error-free programs, which follow best practices, into leaking their secrets. In fact, the safety checks of said best practices actually increase the attack surface and may make applications more susceptible to Spectre.

Spectre is harder to exploit than Meltdown, but it is also harder to mitigate.

Test the Meltdown vulnerability
with these scripts…



Example file for Meltdown

```
240001c9f: | 00 6d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .m.............. |
24000262f: | 00 7d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .}.............. |
24000271f: | 00 00 00 00 00 00 00 00 00 00 00 00 65 6e 20 75 | ............en u |
24000272f: | 73 65 72 20 73 70 61 63 65 20 61 6e 64 20 6b 65 | ser space and ke |
24000273f: | 72 6e 65 6c 57 65 6c 63 6f 6d 65 20 74 6f 20 74 | rnelWelcome to t |
24000298f: | 00 61 72 79 20 62 65 74 77 65 65 6e 20 75 73 65 | .ary between use |
24000299f: | 72 20 73 70 61 63 65 20 61 6e 64 20 6b 65 72 6e | r space and kern |
2400029af: | 65 6c 42 75 72 6e 20 61 66 74 65 72 20 72 65 61 | elBurn after rea |
2400029bf: | 64 69 6e 67 20 74 68 69 73 20 73 74 72 69 6e 67 | ding this string |
240002dcf: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 c8 | ................ |
2400038af: | 6a 75 73 74 20 73 70 69 65 64 20 6f 6e 20 61 00 | just spied on a. |
240003c8f: | 00 00 1e 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................ |
24000412f: | 00 00 00 00 00 00 00 00 00 00 00 00 65 74 73 2e | ............ets. |
24000413f: | 2e 2e 57 65 6c 63 6f 6d 65 20 74 6f 20 74 68 65 | ..Welcome to the |
2400042ff: | 00 00 00 00 00 00 00 00 00 6e 67 72 61 74 75 6c | .........ngratul |
24000430f: | 61 74 69 6f 6e 73 2c 20 79 6f 75 20 6a 75 73 74 | ations, you just |
24000431f: | 20 73 70 69 65 64 20 6f 6e 20 61 6e 20 61 70 70 |  spied on an app |
```

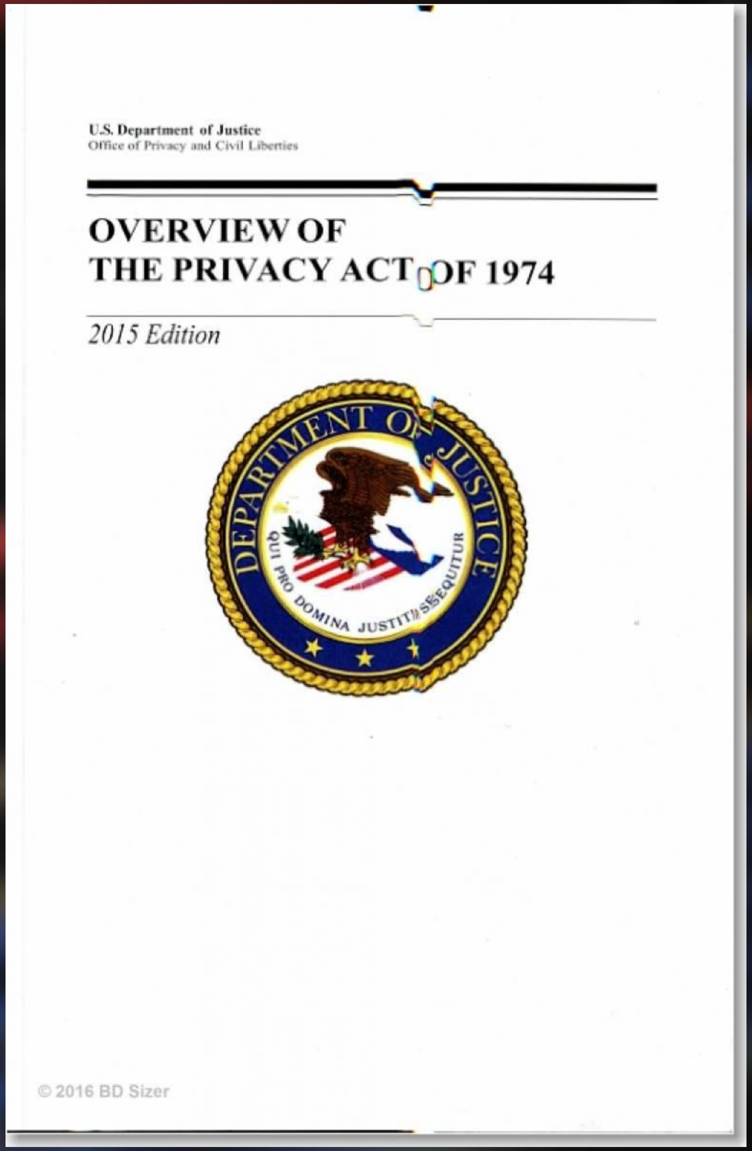
…or look at the official docs.

US Privacy Act of 1974

- While it predates the Internet, the Privacy Act of 1974 probably underpins many Internet and data privacy laws in the United States.

- The act was passed in recognition of the amount of personal data held in computer databases by US government agencies.

- However, the invention of the Internet changed the definition of privacy and made it necessary to enact new data security laws regarding electronic communications.



U.S. Department of Justice
Office of Privacy and Civil Liberties

OVERVIEW OF
THE PRIVACY ACT OF 1974

2015 Edition

© 2016 BD Sizer

Federal Trade Commission Act

- The Federal Trade Commission Act of 1914 was formulated to outlaw unfair methods of competition and unfair acts or practices affecting commerce.

- Today, while the FTC doesn't explicitly regulate what information should be included in website privacy policies, it does use its authority to issue regulations, enforce privacy laws, and protect consumers.

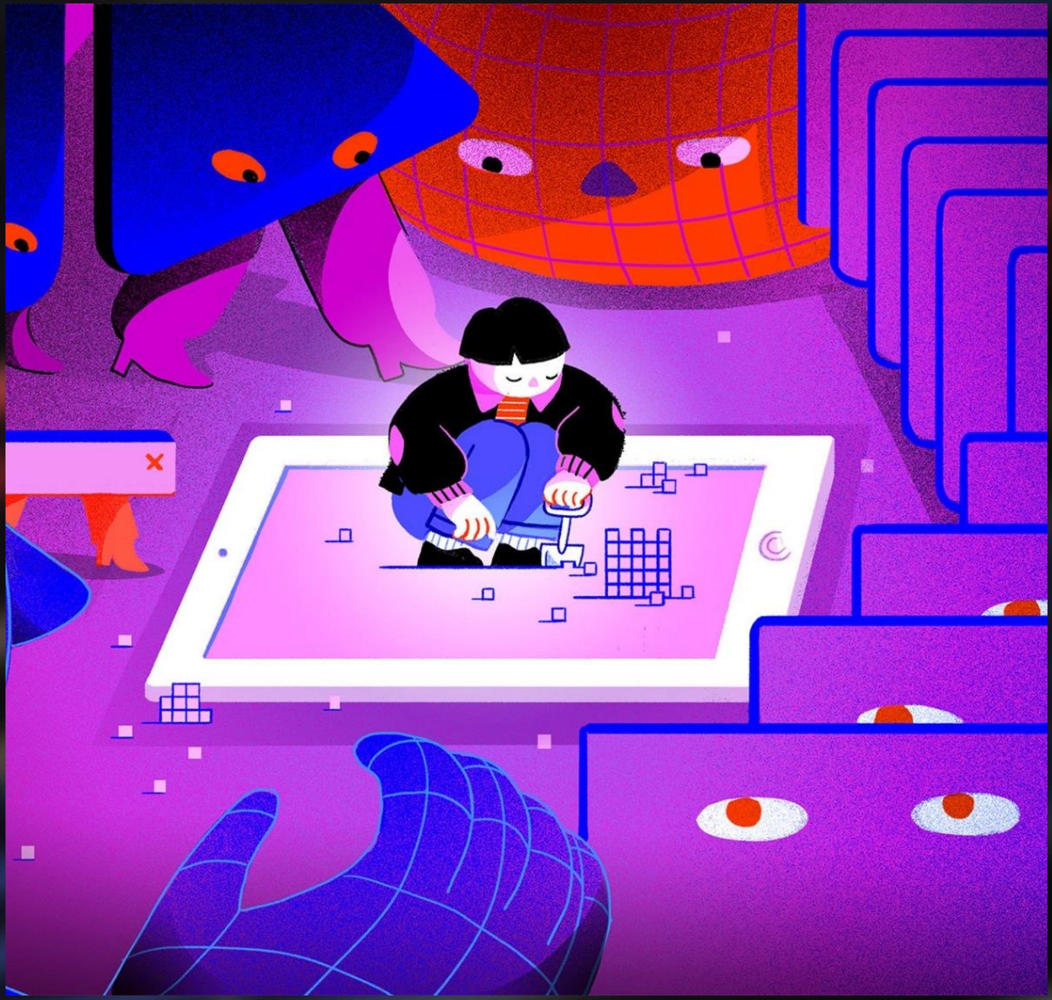
- The FTC plays a role in regulating the Internet, not least because it looks into misleading claims made by major technology and social media companies regarding the privacy of the consumer data they collect.

Children's Online Privacy Protection Act

- Issued in 1998, it is a United States federal law designed to give parents control over what information is collected from their children.

- COPPA applies to operators of commercial websites and online services (including mobile apps and Internet of Things devices) targeting children under 13 that collect personal information from children.

- Websites, apps and online tools aimed at children under 13 should display a warning and obtain parental consent before collecting information from children. They must have a clear and complete privacy policy, and they must ensure the utmost confidentiality of all information obtained from children.



General Data Protection Regulation

- The EU General Data Protection Regulation (GDPR) came into force in 2018. It is a legal arrangement that establishes guidelines for the collection and processing of personal data of people living in the EU.

- The GDPR applies regardless of where websites are based and should therefore be respected by all websites that are visited by European citizens. The GDPR is considered one of the strictest data security laws in the world.

General Data Protection Regulation

- The GDPR specifies that website users must be informed about the data that a site collects and that users must give their explicit consent to data collection. For this reason, many websites display pop-ups asking users to consent to the collection of cookies, which are small files containing personal information such as site settings and preferences.

- The GDPR is detailed on the official website of the European Commission. Some fines imposed on large companies for violations of the GDPR have attracted attention.

Europe Strikes Back

- Google was fined $57 million for hiding important information from users when setting up Android phones, so they didn't know what data collection policies they were agreeing to.

- British Airways was instead fined $28 million when 500,000 customer booking records were stolen in an attack.

- Amazon's Luxembourg EU headquarters was found to be tracking user data without acquiring appropriate consent from users or providing the means to opt out from this tracking — resulting in Amazon being hit with the largest GDPR fine to date (€746 million).

BRITISH AIRWAYS

Google

amazon

THE END