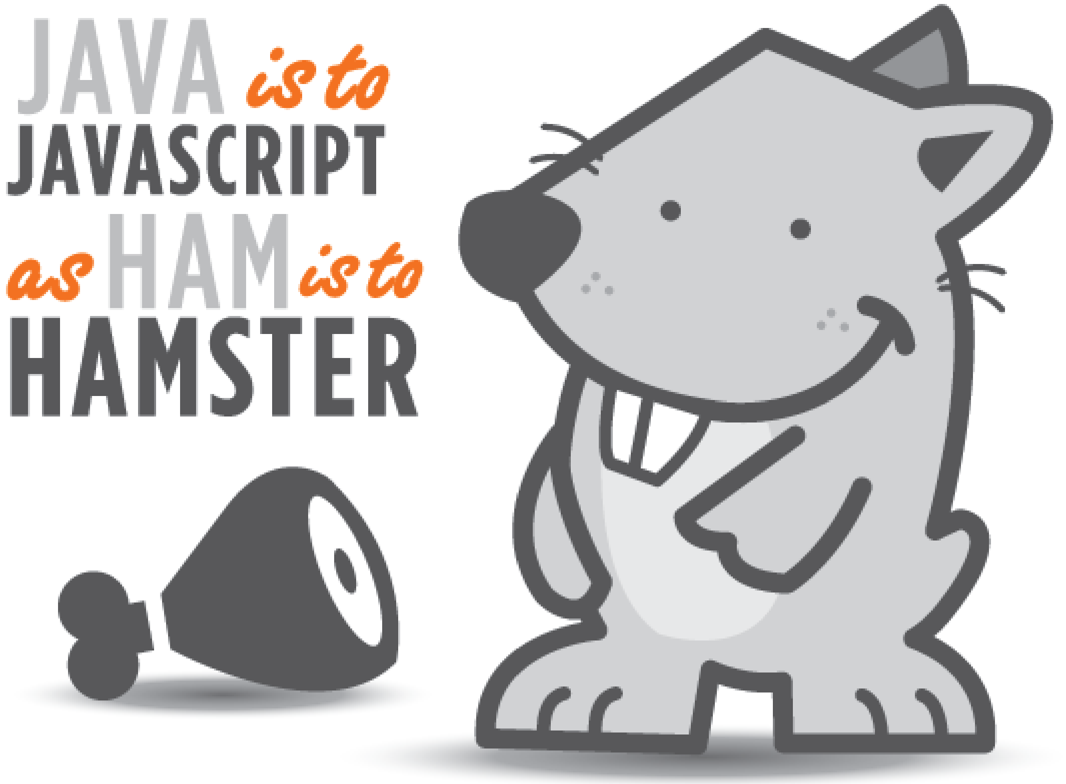
64-31.1 Projet de technologie web

JavaScript

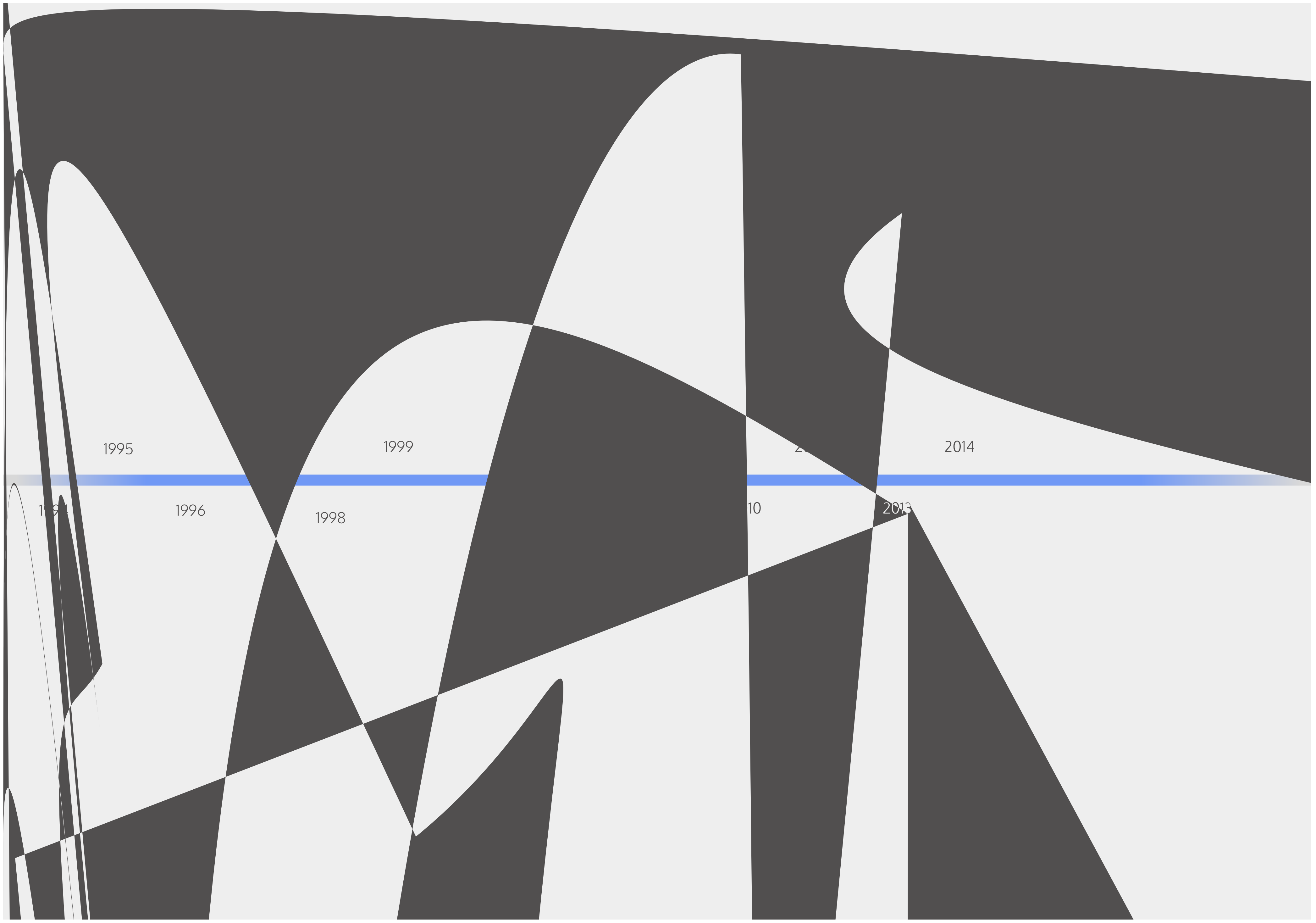
Boris.Fritscher@he-arc.ch

JavaScript



ILLUSTRATED BY SEGUE TECHNOLOGIES

The ECMAScript Standard Timeline



1995
1999
2014
1994
1996
1998
10
2013

JavaScript

JavaScript is an important language because it is the language of the web browser. Its association with the browser makes it one of the most popular programming languages in the world. **At the same time, it is one of the most despised programming languages in the world**. [...]

Most people in that situation **don't even bother to learn JavaScript first**, and then they are surprised when JavaScript turns out to have significant differences from the some other language they would rather be using, and that those differences matter.

The amazing thing about JavaScript is that it is possible to get work done with it without knowing much about the language, or even knowing much about programming. It is a language with enormous expressive power. It is even better when you know what you're doing. **Programming is difficult business. It should never be undertaken in ignorance.**

JavaScript: The Good Parts -- Douglas Crockford

Douglas Crockford: JavaScript: The Good Parts

https://www.youtube.com/watch?v=_DKkVvOt6dk

Douglas Crockford: JavaScript: The Good Parts

JavaScript 101 (Part 1)

- Variables / Constants
- Types
- Template Strings
- Operators
- Arrays
- Loops
- Conditions
- Functions
- Objects

Variables / Constants / Comments

```
// This is a single line comment

// block variables use let over var!
let bar;
bar = 'hello';
let baz = 'world';

// variables before ES2015
// scope is only at the function level
var foo;

// constants
const cannotBeReassigned = 'The One';

/*
  This is a multi-line comment. It can go on
  for several lines, like this.
 */
```

Types

JavaScript defines **6 types**:

- number
- boolean
- string
- object
- undefined
- null

```
let aNumber = 3.12;
let aBoolean = true;
let aString = 'John Smith';
let anObject = { aProperty: null };
typeof aNumber === 'number';
typeof aBoolean === 'boolean';
typeof aString === 'string';
typeof anObject === 'object';
typeof anObject.aProperty === 'object';
typeof anObject.foobar === 'undefined';
// null is a type but
typeof null === 'object';
```

JavaScript is a dynamic language: when you declare a variable, you don't specify a type (and the type can change over time).

Template Strings

New in ES2015 in addition to '', "", there are ``.

```
// Basic literal string creation
const s1 = `This is a pretty little template string.`;

// Multiline strings
const s2 = `In ES5 this is
 not legal.`;

// Interpolate variable bindings
let name = "Bob", time = "today";
const s3 = `Hello ${name}, how are you ${time}?`;
```

Operators

Operator	Example
+	2 + 5 === 7 'H' + 3 === 'H3'
-	5 - 3 === 2
==	Returns true if the operands are equal. 3 == var3 "3" == var3 3 == '3'
!=	Returns true if the operands are not equal.
===	Returns true if the operands are equal and of the same type.
!==	Returns true if the operands are of the same type but not equal, or are of different type.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators

Array

```
let myArray = [ 'a', 'b', 'c' ];
let firstItem = myArray[ 0 ];
let secondItem = myArray[ 1 ]; // access the item at index 1
let arrayLength = myArray.length;
```

Arrays are objects

```
const fruits = ['apple', 'pear'];
console.log(typeof fruits); //object

// add elements to an array
fruits.push('banana');

// check if an array contains an element
const inArray = fruits.indexOf('banana') > -1;
const inArray = fruits.includes('banana'); // new in ES6

// remove 1 element from array
const removed = fruits.splice(fruits.indexOf('pear'), 1);

const length = fruits.length;
```

Loop

```
//iterate over an array
for (let i = 0; i < fruits.length; i++) {
    console.log( 'fruit at index ' + i + ' is ' + fruits[ i ] );
}
```

```
let i = 0;
while (i < myArray.length) {
  console.log( `item at index ${i} is ${myArray[ i ]}` );
  i++;
}
```

```
for (let value of array) {
  // do something with value
}
```

```
for (let property in object) {
  // do something with object property
}
```

Condition

```
let name = "kittens";
if (name === "puppies") {
  name += "!";
} else if (name === "kittens") {
  name += "!!";
} else {
  name = "!" + name;
}

name === "kittens!!"

// ternary
const  result = condition ? expression_if_true : expression_if_false
```

Functions

```
function add(a, b) {
  const total = a + b;
  return total;
}
add(); // NaN
// You can't perform addition on undefined

add(2, 3, 4); // 5
// added the first two; 4 was ignored

const addTwoNumbers = function(a, b) {
  return a + b;
};
addTwoNumbers // return the function
addTwoNumbers(2, 2) // === 4 // function call returns function result
```

Arrow functions

```
const hello = () => {
  return 'world';
}

// Parentheses are optional when there's only one parameter name:
(singleParam) => { statements }
singleParam => { statements }

const addTwoNumbers = (a, b) => a + b;
```

An arrow function does not create its own this context, so this has its original meaning from the enclosing context. => this behaves more like you might think

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Fonctions/Fonctions_fl%C3%A9ch%C3%A9es

Objects are dynamic bags of properties

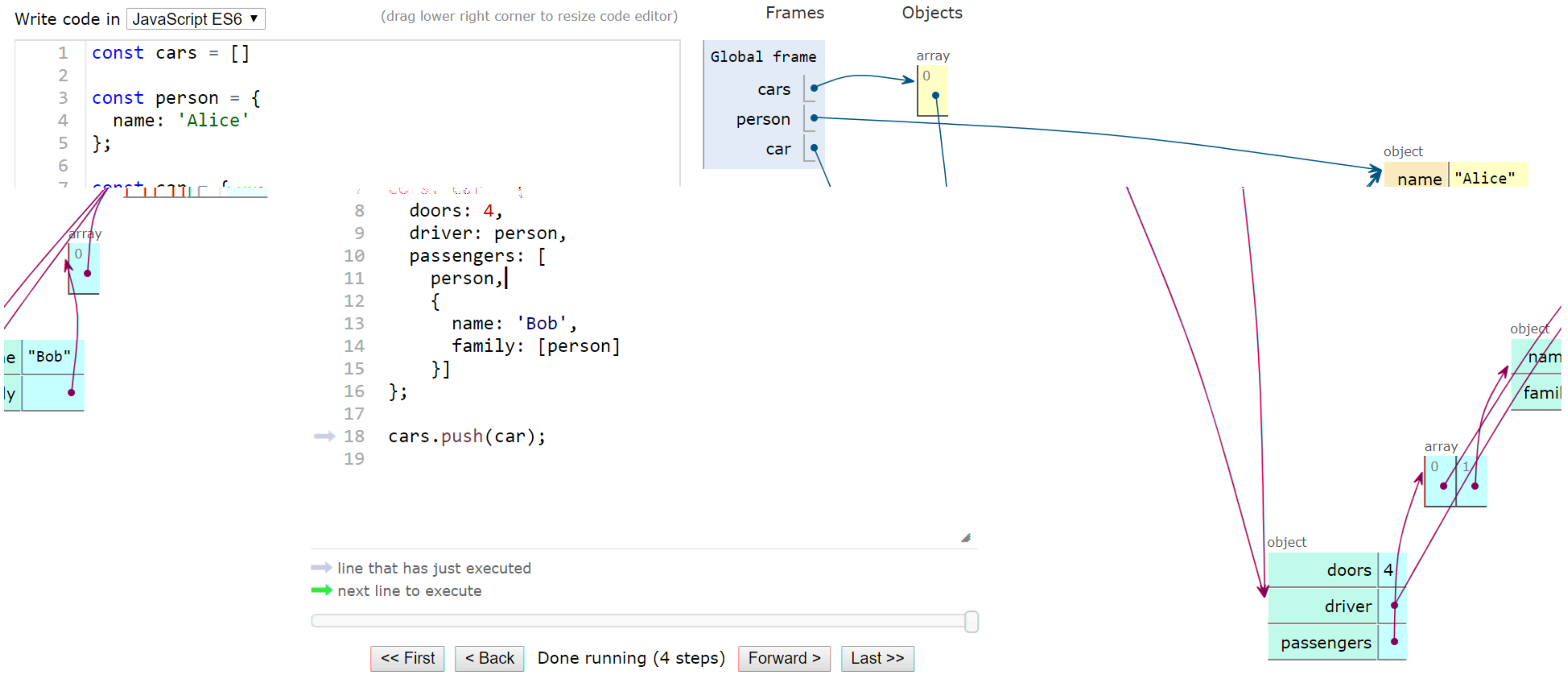
```javascript
// create an object
const person = {
    firstName: 'John',
    lastName: 'Smith'
};
// dynamically add/remove properties
person.gender = 'male';
person['zip'] = 2000;
const key = 'height';
person[key] = 170;

delete person.zip;

// check existence of a property
person.hasOwnProperty('gender');

// enumerate properties
for (const key in person) {
    console.log(key + ' : ' + person[key]);
}
```

There are different ways to **access properties** of an object.

JavaScript is **dynamic**: it is possible to **add** and **remove** properties to an object at any time.

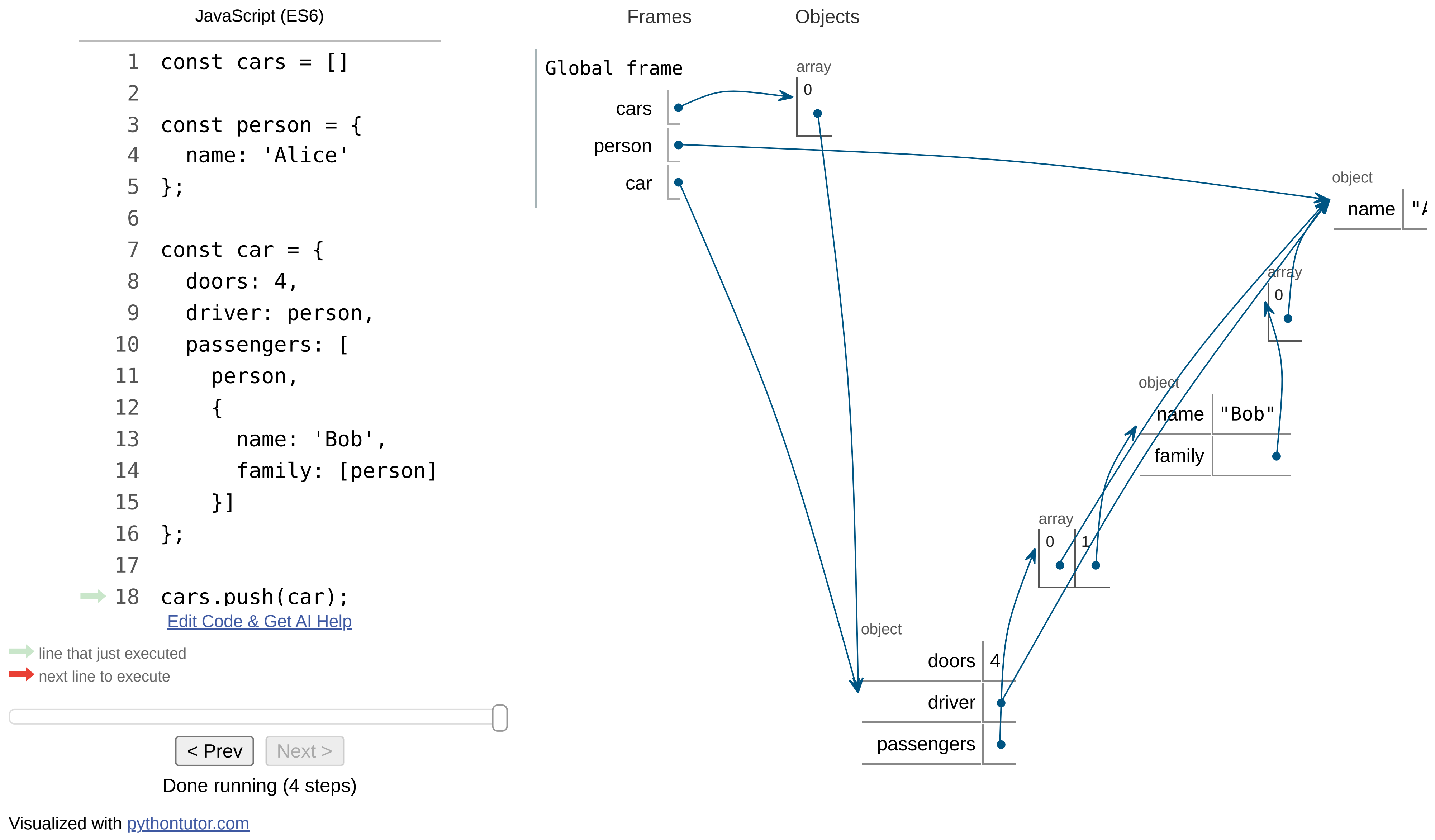
Every object has a different list of properties (**no class**).

Visualize Variables



```
const cars = []

const person = {
  name: 'Alice'
};

const car = {
  doors: 4,
  driver: person,
  passengers: [
    person,
    {
      name: 'Bob',
      family: [person]
    }]
};

cars.push(car);
```

https://pythontutor.com/visualize.html#mode=edit



JavaScript (ES6)

```
const cars = []

const person = {
  name: 'Alice'
};

const car = {
  doors: 4,
  driver: person,
  passengers: [
    person,
    {
      name: 'Bob',
      family: [person]
    }]
};

cars.push(car);
```

Edit Code & Get AI Help

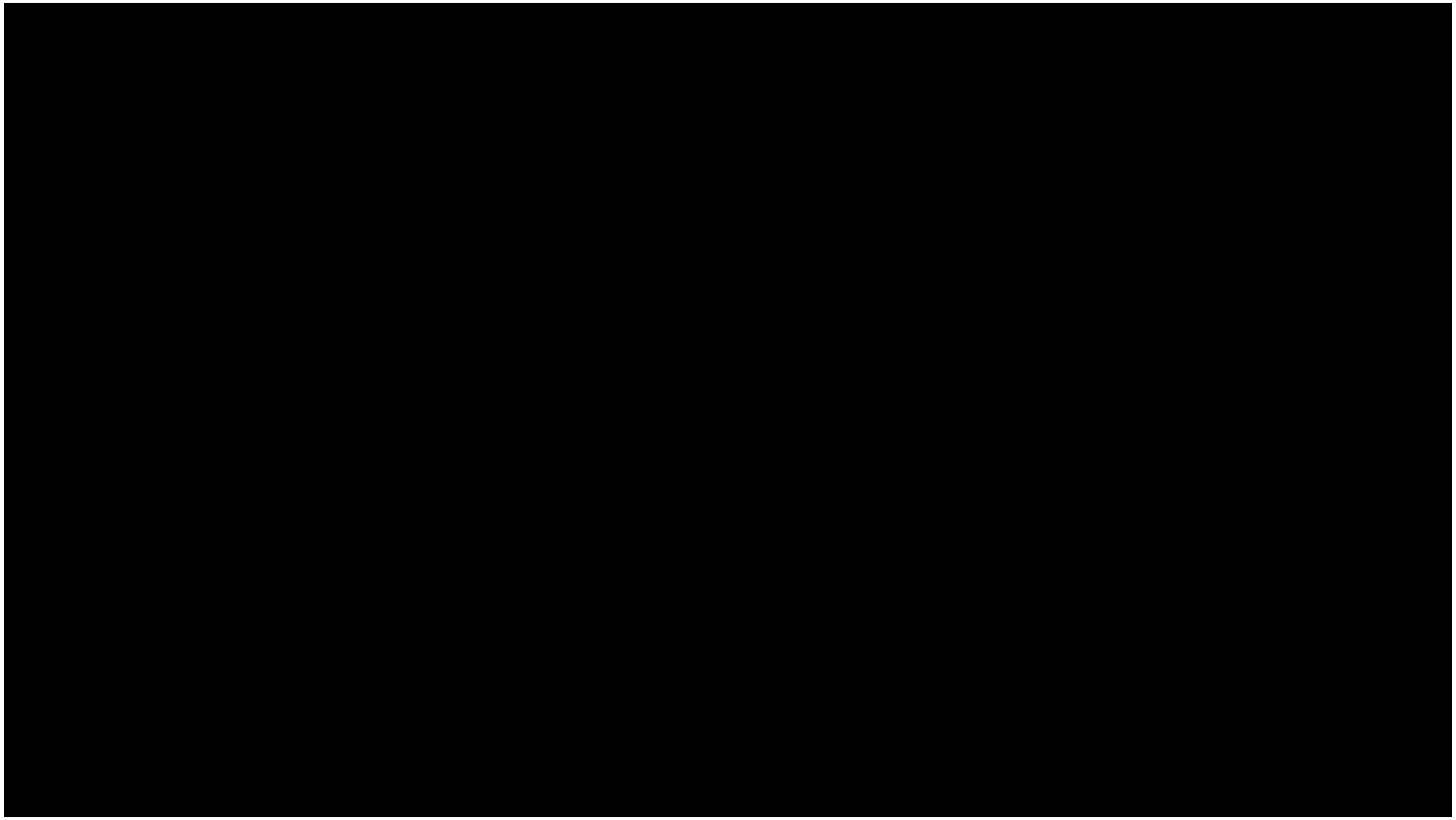
line that just executed
next line to execute

< Prev Next >

Done running (4 steps)

Visualized with pythontutor.com

JavaScript WAT



http://stackoverflow.com/questions/9032856/what-is-the-explanation-for-these-bizarre-javascript-behaviours-mentioned-in-the

https://www.destroyallsoftware.com/talks/wat

Assignment AlgoJS

1. Generate a github repository for you by clicking here
2. Clone the project in VS Code
3. `npm install` dependencies
4. Install Mocha Test Explorer for VSCode
5. Write code to pass the tests
6. Commit and push your changes
7. Check your status on https://pweb.bf0.ch/

JavaScript 101 (Part 2)

- Classes
- Arrays the functional way
- DOM

Classes since ECMAScript 2015

```
class SkinnedMesh extends THREE.Mesh {
  constructor(geometry, materials) {
    super(geometry, materials);

    this.idMatrix = SkinnedMesh.defaultMatrix();
    this.bones = [];
    this.boneMatrices = [];
    //...
  }
  update(camera) {
    //...
    super.update();
  }
  static defaultMatrix() {
    return new THREE.Matrix4();
  }
}
const mesh = new SkinnedMesh(geometry, materials);
```

Arrays the functional way

```javascript
const fruits = ['abricot', 'ananas', 'strawberry', 'orange'];

// creates a new array with the results of calling a provided function
// on every element in the calling array.
const transformedFruits = fruits.map(fruit => {
    return fruit.toUpperCase();
});

// executes a provided function once for each array element.
transformedFruits.forEach(fruit => {
    console.log(fruit);
});

// creates a new array with all elements that pass the test implemented
// by the provided function.
const aFruits = fruits.filter(fruit => {
    return fruit.charAt(0) === 'a';
});
```

Arrays the functional way

```
const fruits = ['abricot', 'ananas', 'strawberry', 'orange'];

// executes a reducer function (that you provide)
// on each element of the array,
// resulting in a single output value.

const count = fruits.reduce((val, fruit) => {
    console.log('reducer invoked with ' + val);
    return val + 1;
}, 0);
console.log('There are ' + count + ' fruits in the array');
```

There are more functional methods: sort, some, every, flat, flatMap
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

ECMAScript 2015-2016

- Module
- Enhanced Object Literals
- Destructuring
- Default + Rest + Spread

Modules since ECMAScript 2015

import/export functions and variables between files.

```
// lib/math.js
export function sum(x, y) {
  return x + y;
}
export const pi = 3.141593;
```

```
// app.js
import * as math from "lib/math";
console.log("2π = " + math.sum(math.pi, math.pi));
```

import/export syntax

```
import defaultMember from "module-name";
import * as name from "module-name";
import { member } from "module-name";
import { member as alias } from "module-name";
import { member1 , member2 } from "module-name";
import { member1 , member2 as alias2 , [...] } from "module-name";
import defaultMember, { member [ , [...] ] } from "module-name";
import defaultMember, * as name from "module-name";
import "module-name";
```

```
export { name1, name2, …, nameN };
export { variable1 as name1, variable2 as name2, …, nameN };
export let name1, name2, …, nameN; // also var
export let name1 = …, name2 = …, …, nameN; // also var, const

export default expression;
export default function (…) { … } // also class, function*
export default function name1(…) { … } // also class, function*
export { name1 as default, … };

export * from …;
export { name1, name2, …, nameN } from …;
export { import1 as name1, import2 as name2, …, nameN } from …;
```

Destructuring

```
// list matching
const [a, ,b] = [1,2,3];
a === 1;
b === 3;

// Fail-soft destructuring
const [a] = [];
a === undefined;

// Fail-soft destructuring with defaults
[a = 1] = [];
a === 1;

// object matching
const {name: n, likes: [,,c]} = {name: 'hello', likes: ['cat', 'dog', 'cow']};
n === 'hello';
c === 'cow';
```

Default + Rest + Spread

```
function f(x, y=12) {
  // y is 12 if not passed (or passed as undefined)
  return x + y;
}
f(3) == 15

function f(x, ...y) {
  // y is an Array
  return x * y.length;
}
f(3, "hello", true) == 6

function f(x, y, z) {
  return x + y + z;
}
// Pass each elem of array as argument
f(...[1,2,3]) == 6
```

ECMAScript 2015-2016

And a lot more:

- Iterators
- Generators
- Unicode
- Map, Set, WeakMap, WeakSet
- Proxies
- Symbols
- Async Await
- Tail Calls

https://github.com/DrkSephy/es6-cheatsheet
https://babeljs.io/learn-es2015/

DOM Document

```javascript
// access body Element
const element = document.body;

// find element(s)
const element = document.getElementById("some_id");

// returns an Element from a CSS Selector
const parentElement = document.querySelector("ul");

// returns an iterable of Elements from a CSS Selector
const elements = document.querySelectorAll("ul.someClass > li");
const element = elements[0];

// create a new element
const element = document.createElement("div");

// add to DOM
parentElement.append(element);

// Page ready
document.addEventListener('DOMContentLoaded', function() {});
```

https://developer.mozilla.org/en-US/docs/Web/API/Element

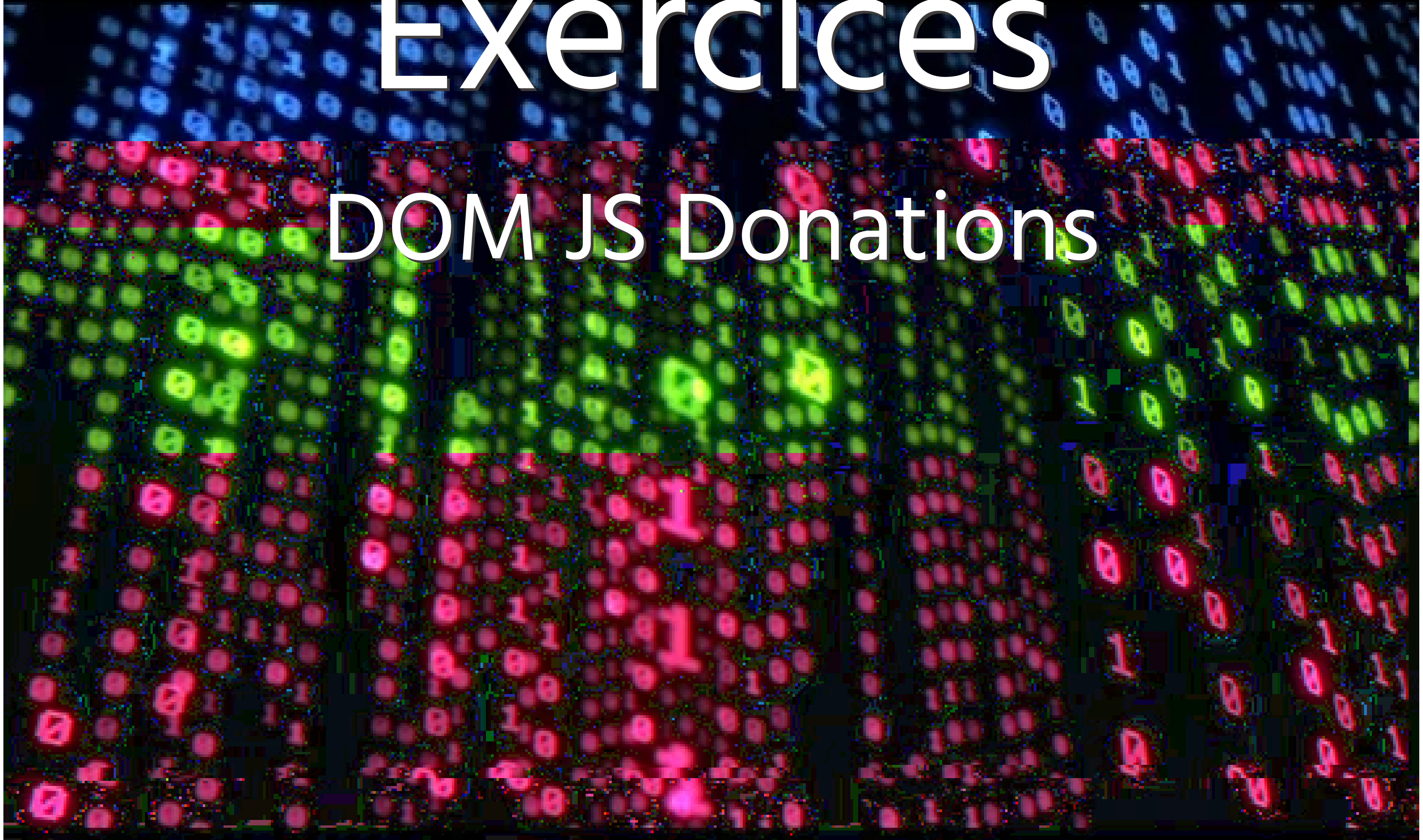
DOM Element

```javascript
// edit classes
element.classList.add("big");
element.classList.remove("big");
element.classList.toggle("big");

// edit content
const value = element.innerText;
element.innerText = "some text";
element.innerHTML = "text with <b>HTML</b>";

// edit attributes
const value = element.getAttribute("src");
element.setAttribute("src", "https://...");

// events: click, dblclick, change, keydown, mouseenter, mouseleave
element.addEventListener('click', () => {
  // handler function
  // do something
});
```

```html
<!DOCTYPE html>
<html>
<body onload="onload()">
  <ul class="list">
    <li>a</li>
    <li>b</li>
  </ul>
</body>
</html>
```

```css
.big{
  font-size: 200%;
}
```

```javascript
function onload() {
  const div = document.createElement('div');
  div.innerText = 'Hello';
  document.body.append(div);

  for (const li of document.querySelectorAll('.list li')) {
    li.addEventListener('click', () => {
      li.classList.toggle('big');
    })
  }
}
```



Exercices

DOM JS Donations

References

- https://developer.mozilla.org/fr/docs/Web/JavaScript/Une_r%C
- https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide
- https://developer.mozilla.org/en-US/Learn/Getting_started_wit
- http://sutterlity.gitbooks.io/apprendre-jquery/content/rappel_j
- http://eloquentjavascript.net/
- https://developer.chrome.com/devtools
- https://babeljs.io/learn-es2015/