

JavaScript.

The Fundamentals

JavaScript - Behavior structures

JavaScript functions.

- **Fundamental unit of composition for logic (or BEHAVIOUR).**
- **Function syntax:**
 - **ES5:**
 - **Function declarations.**
 - **Function expressions.**
 - **Hoisting (ES5) – all function declarations moved to the top of the current scope at runtime – now redundant.**
 - **ES6:**
 - **Arrow functions.**
 - **Shorthand version.**
 - **Anonymous functions (see later).**
- **Ref. functions/01_functionBasics.js**

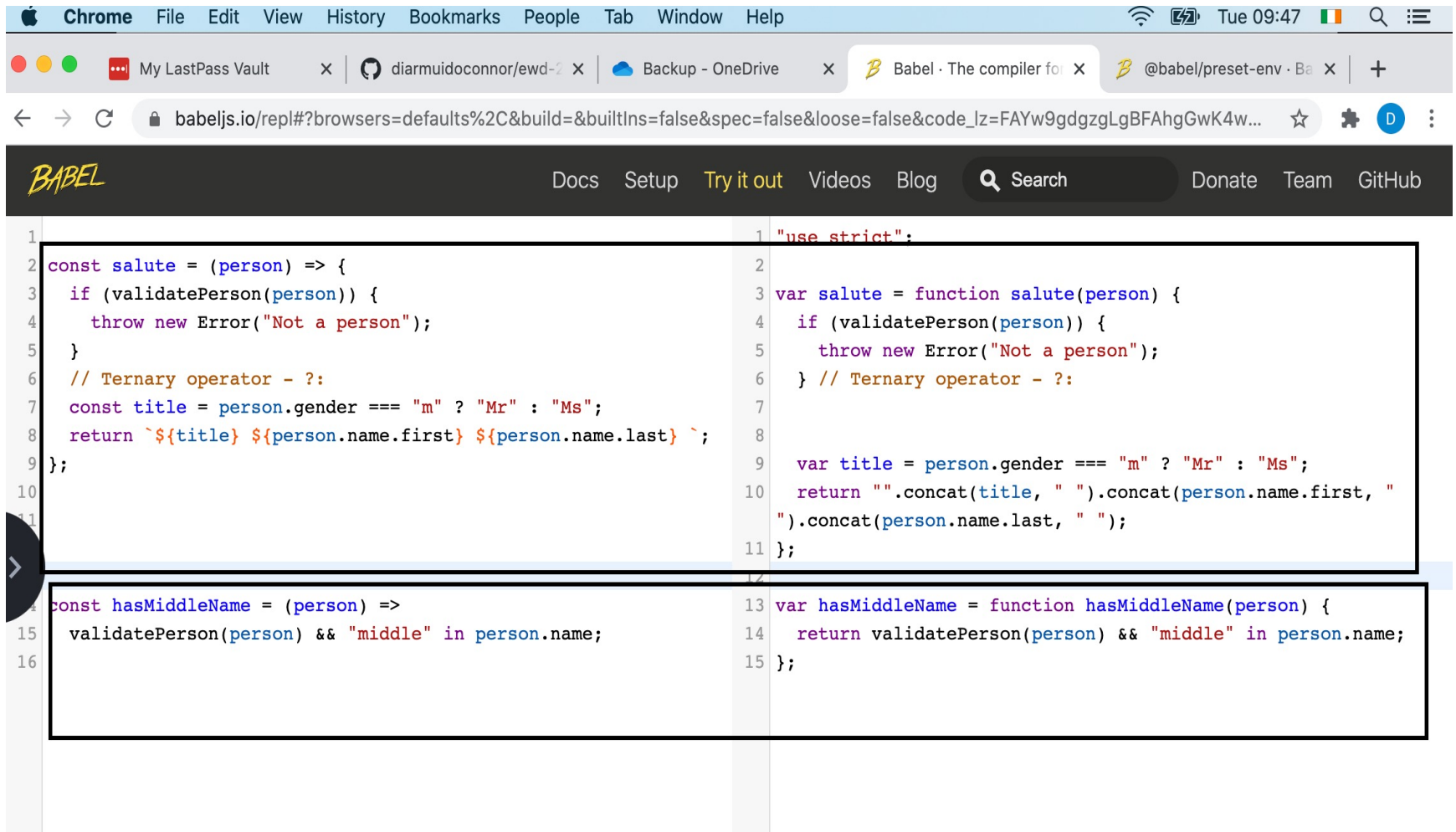
Arrow functions

- **A cleaner syntax for creating functions.**

const name = (parameters) => { Body }

- **The => (arrow) separates function body from its parameters.**
- **Enclose body with curly braces, { }.**
 - **Unless body is a single expression (optional).**
- **Enclose parameter list with parentheses, (...).**
 - **Unless only a single parameter (optional).**
- **Omit return token when single-expression body (optional).**

Arrow functions – ES6 → ES5



The screenshot shows the Babel REPL interface in a Chrome browser. The browser's address bar displays the URL `babeljs.io/repl/#?browsers=defaults%2C&build=&builtIns=false&spec=false&loose=false&code_lz=FAYw9gdgzgLGbFAhgGwK4w...`. The Babel logo is in the top left, and navigation links for Docs, Setup, Try it out, Videos, Blog, Search, Donate, Team, and GitHub are in the top right. The main area is split into two panels: the left panel shows the input ES6 code, and the right panel shows the output ES5 code.

ES6 Input Code:

```
1
2 const salute = (person) => {
3   if (validatePerson(person)) {
4     throw new Error("Not a person");
5   }
6   // Ternary operator - ?:
7   const title = person.gender === "m" ? "Mr" : "Ms";
8   return `${title} ${person.name.first} ${person.name.last}`;
9 };
10
11
12
13
14 const hasMiddleName = (person) =>
15   validatePerson(person) && "middle" in person.name;
16
```

ES5 Output Code:

```
1 "use strict";
2
3 var salute = function salute(person) {
4   if (validatePerson(person)) {
5     throw new Error("Not a person");
6   } // Ternary operator - ?:
7
8   var title = person.gender === "m" ? "Mr" : "Ms";
9   return "".concat(title, " ").concat(person.name.first, "
10 ").concat(person.name.last, " ");
11 };
12
13 var hasMiddleName = function hasMiddleName(person) {
14   return validatePerson(person) && "middle" in person.name;
15 };
16
```

Function characteristics

- Constructor functions – **function for creating objects of a certain type, e.g.**
 function Person(.....) { }
 let him = new Person('joe Bloggs', '1 Main Street',)
 - **Same purpose as classes in Java.**
- Side-effects – **when a function “modifies some state variable value(s) outside of its local environment”.**
 - e.g. **addMiddleName() causes a side-effect.**
 salute() does not cause side-effects.
 - **Performing I/O also considered a side-effect.**
- Pure function – **has no side-effects; will always return the same result for a given set of parameters.**
 - **Functional programming.**

Higher Order Functions (HOF).

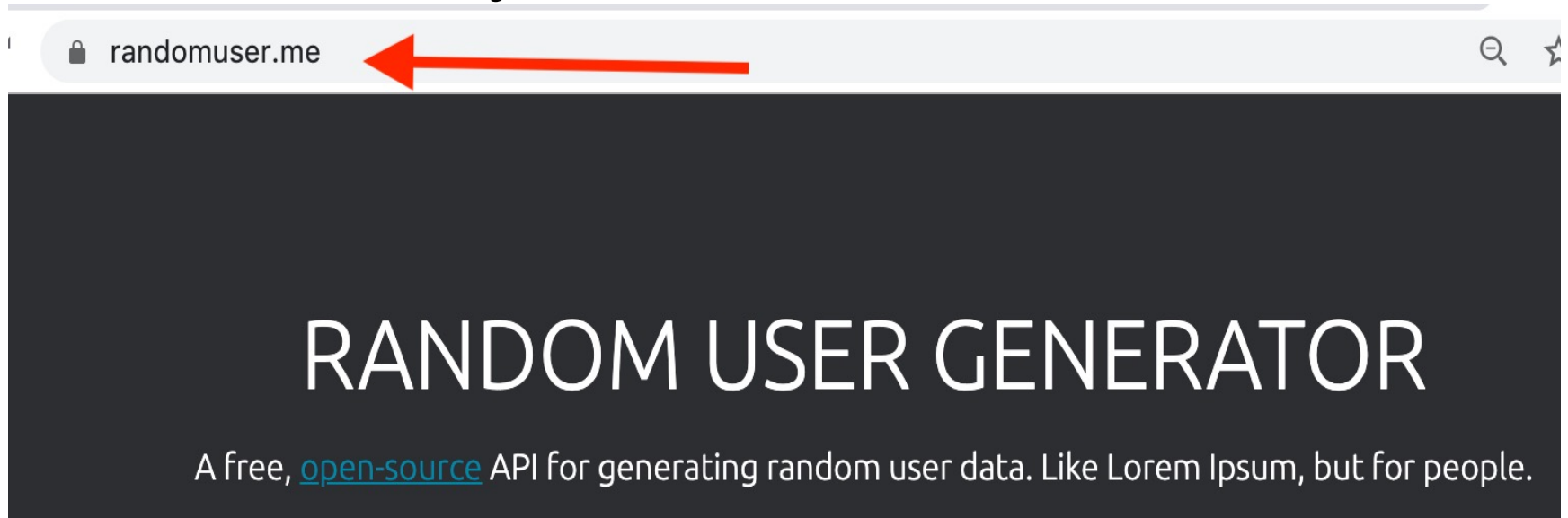
- **Definition: A function that takes a function as a parameter (and/or returns a function response).**
 - **Function parameter termed a callback.**
function someHOF(. . ., callback,)
 - **Callback usually coded as an anonymous function.**
- **Case study – The Array HOFs.**
 - **forEach()**
 - **filter()**
 - **map()**
 - **reduce()**

Array HOFs – forEach().

```
const sourceArray = [ ..... ]  
sourceArray.forEach(  
  function(element, index, array) { ...Anonymous function body ...}  
)
```

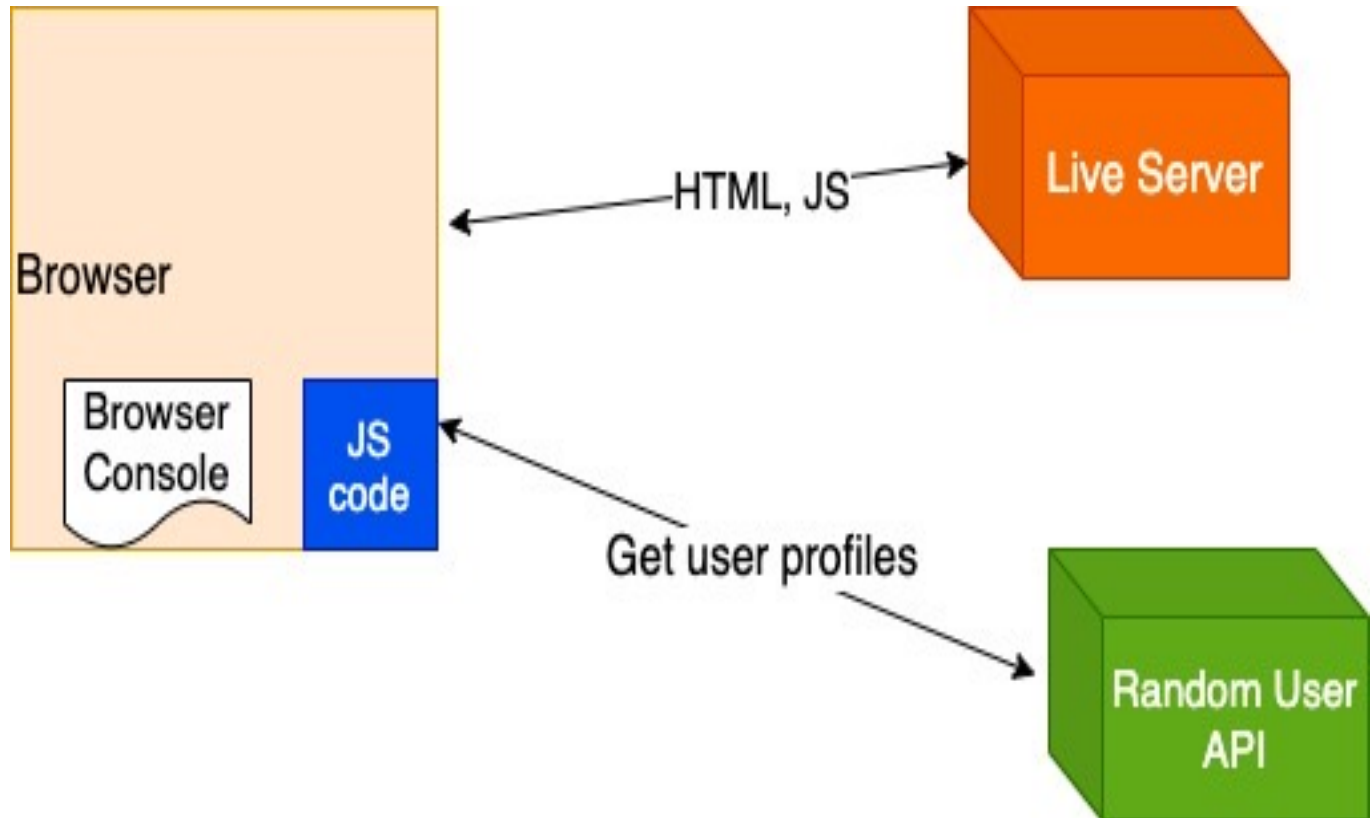
- **Anonymous function is called for each array element.**
- **An alternative to using for-loop.**
- **index and array arguments are optional.**
- **More commonly coded using Arrow function style.**

Array HOF demos context



- **Open Web API.**
- **Accepts HTTP GET requests, e.g.**
<https://randomuser.me/api/?results=10> - generate 10 user profiles and returns them in a JSON (Javascript Object Notation) structure.

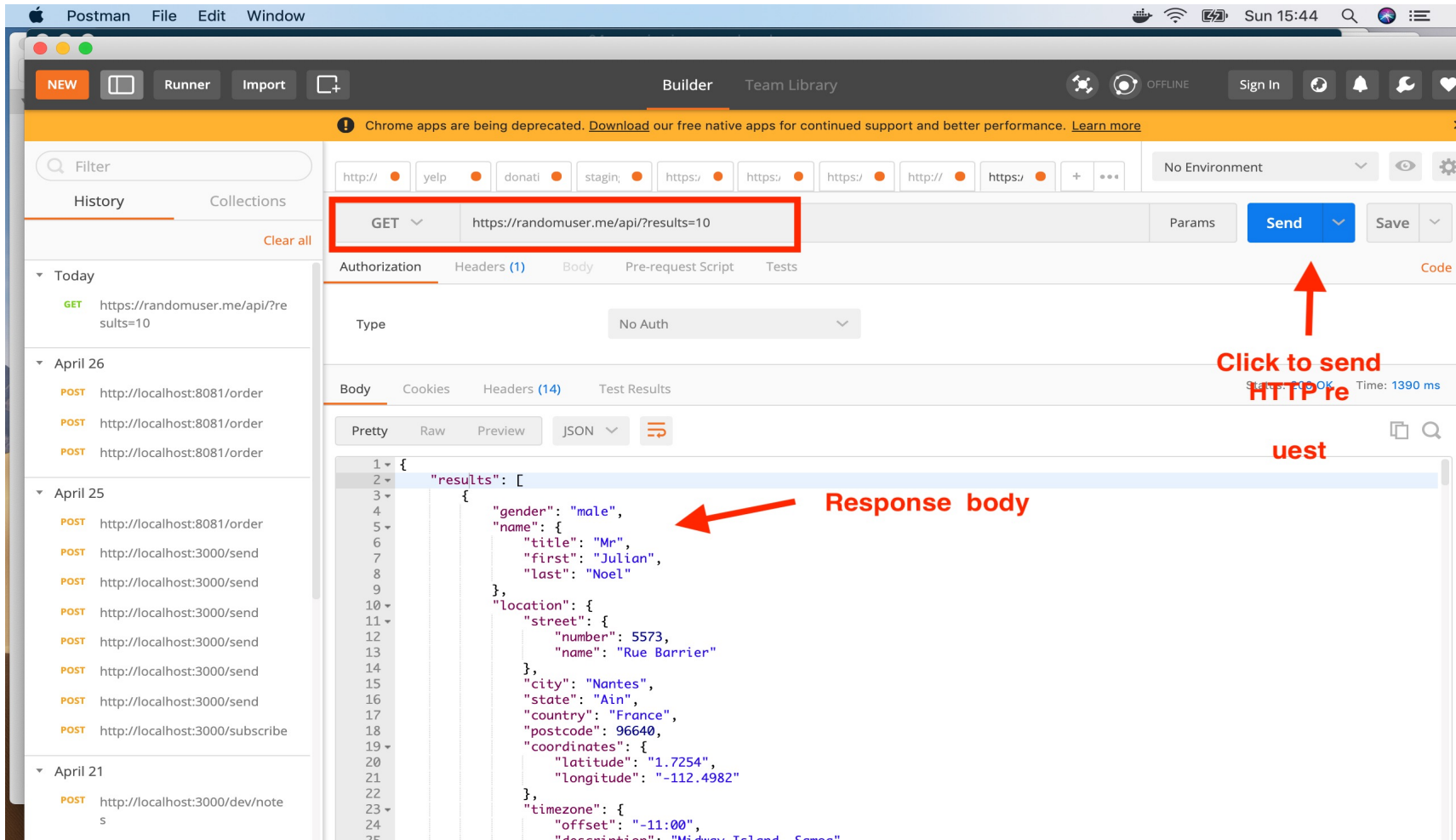
Array HOF demos context



Use Postman to test Web APIs.

(Postman = Chrome extension or app)

- We'll use it later in this module.



Array HOF demos.

- **Base example.**
 - `fetch()` **and** `array.forEach(callback)`
 - **Ref. functions/02_webAPICall.js.**
- *filter(callback).*
 - **Select entries from a source array, based on some criteria.**
 - **Selected entries added to a new array.**
 - **Source array unchanged (Pure).**
 - **Ref. functions/03_filtering.js**
- *map(callback).*
 - **Creates a new array, based on the source – 1-for-1 mapping.**
 - **Source array unchanged (Pure).**
 - **Ref. functions/04_mapping.js**

Array HOF demos.

```
accumulator = sourceArray.reduce(  
  (acc, element, index, array) => {  
    // Callback  
    .....  
    return updatedAccumulator  
  }, initialAccumulator )    // Note .
```

- *reduce(.....)*
 - **reduces the source array to a single accumulated value.**
 - **Source array unchanged (Pure)**
 - **Callback incrementally ‘computes’ the accumulator.**
 - **Accumulator passed between invocations of the callback.**
 - **Ref functions/05_reducing.js**

Summary

- **Defining Behavior.**
 - **Functions:**
 - **ES5 – Function declarations; Function expressions.**
 - **ES6 – Arrow functions. Shorthand.**
 - **Anonymous functions.**
 - **Higher Order functions.**

