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. archive -** functions/01_functionBasics.js

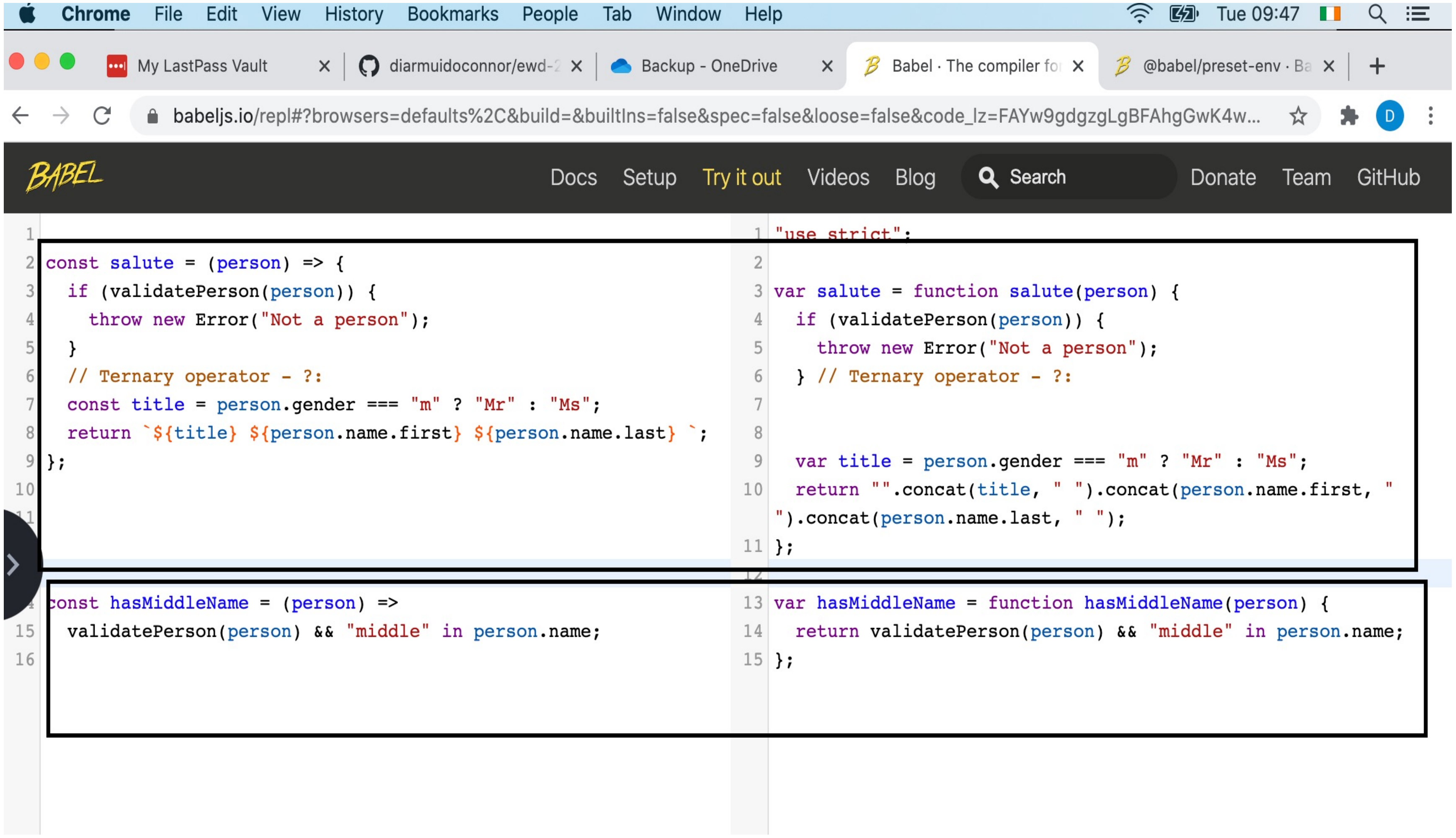
Arrow functions

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

  const name = (parameters) => { ….. Body ……. }

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

Arrow functions – ES6 → ES5


Chrome File Edit View History Bookmarks People Tab Window Help
Tue 09:47
My LastPass Vault
diarmuidoconnor/ewd-2
Backup - OneDrive
Babel · The compiler fo
@babel/preset-env · Ba
babeljs.io/repl#?browsers=defaults%2C&build=&builtIns=false&spec=false&loose=false&code_lz=FAYw9gdgzgLgBFAhgGwK4w...
BABEL
Docs
Setup
Try it out
Videos
Blog
Search
Donate
Team
GitHub

```
const salute = (person) => {
  if (validatePerson(person)) {
    throw new Error("Not a person");
  }
  // Ternary operator - ?:
  const title = person.gender === "m" ? "Mr" : "Ms";
  return `${title} ${person.name.first} ${person.name.last} `;
};


const hasMiddleName = (person) =>
  validatePerson(person) && "middle" in person.name;
```

```
"use strict";

var salute = function salute(person) {
  if (validatePerson(person)) {
    throw new Error("Not a person");
  } // Ternary operator - ?:


  var title = person.gender === "m" ? "Mr" : "Ms";
  return "".concat(title, " ").concat(person.name.first, " ").concat(person.name.last, " ");
};

var hasMiddleName = function hasMiddleName(person) {
  return validatePerson(person) && "middle" in person.name;
};
```

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 class constructur in Java.**
- Side-effects **– when a function "**modifies some state variable 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, ….) {…… Body ……}
  - **Callback is usually an** anonymous **function.**

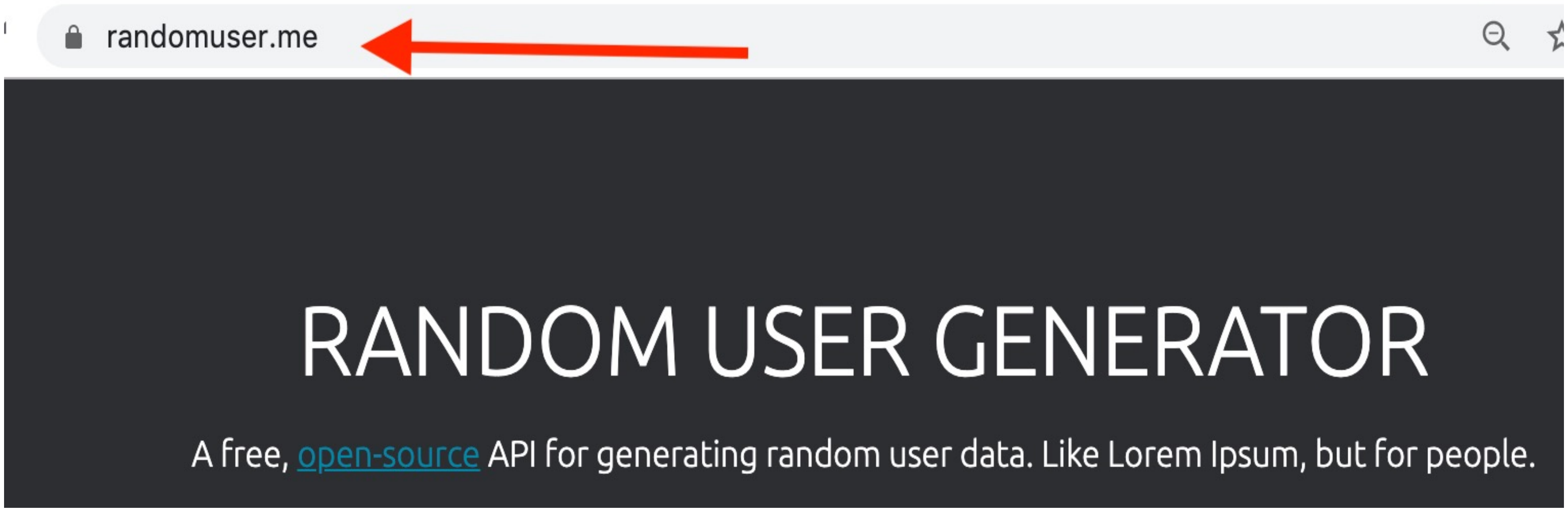
- **Case study – The Array HOFs.**
  - **forEach()**
  - **filter()**
  - **map()**
  - **reduce()**

Array HOFs – forEach().

```
const sourceArray = [el1, el2, …..]
sourceArray.forEach(
  function(element, index, array) { …body…} // Anonymous function
)
```

- forEach **executes the anonymous function for each array element.**
- forEach **is an alternative to a for-loop.**
- index **and** array **arguments are optional.**
- **More commonly coded using Arrow function style.**

Array HOF demos – Data source



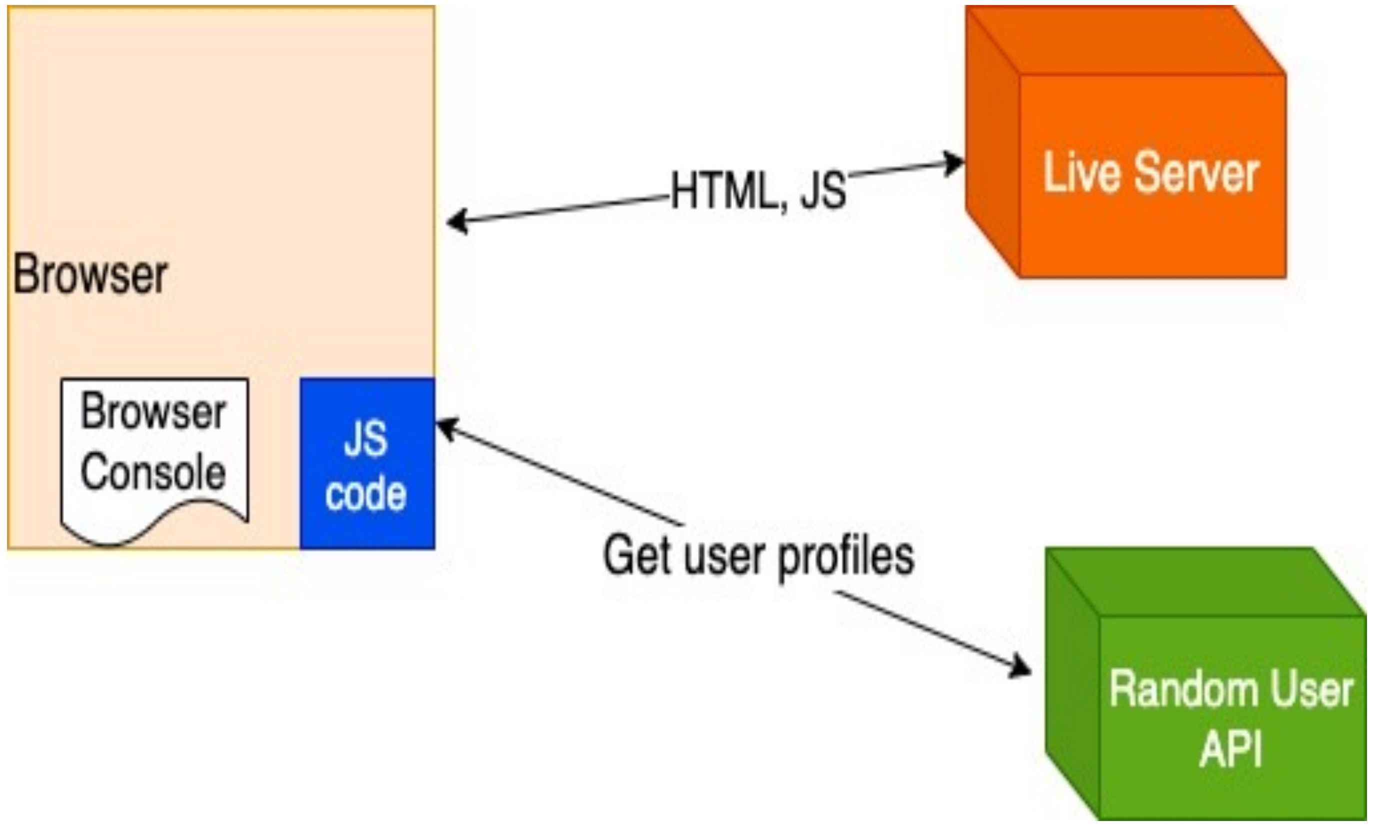
RANDOM USER GENERATOR

A free, open-source API for generating random user data. Like Lorem Ipsum, but for people.

- **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 - Architecture



Array HOF demos - Code.

- **Base example.**
  - fetch() **and** array.forEach(callback)
  - **Ref.** functions/02_webAPICall.js**.**
- **filter(callback).**
  - **Select a subset of elements from a source array.**
  - **Selected element references added to a <u>new</u> 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)**
  - **The callback incrementally ‘builds’ the accumulator.**
  - **Accumulator is passed between callback invocations.**
  - **Ref** functions/05_reducing.js

Summary

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