TABLE OF CONTENTS: THEORY LECTURES (CLICK THE TITLES)

SECTION 1 — INTRODUCTION

WHAT IS JAVASCRIPT?

- JavaScript is a lightweight, cross-platform, object-oriented computer programming language 😅
- JavaScript is one of the three core technologies of web development
- Today, JavaScript can be used in different places:
  - **Client-side: JavaScript was traditionally only used in the browser**
  - Server-side: Thanks to node.js, we can use JavaScript on the server as well
- Javascript is what made modern web development possible:
  - Dynamic effects and interactivity
  - Modern web applications that we can interact with
- **Frameworks/libraries like React and Angular are 100% based on JavaScript: you need to master JavaScript in order to use them!**

02 JavaScript Language Basics

03 How JavaScript Works Behind the Scenes

04 JavaScript in the Browser: DOM Manipulation and Events

05 Advanced JavaScript: Objects and Functions

06 Putting It All Together: The Budget App Project

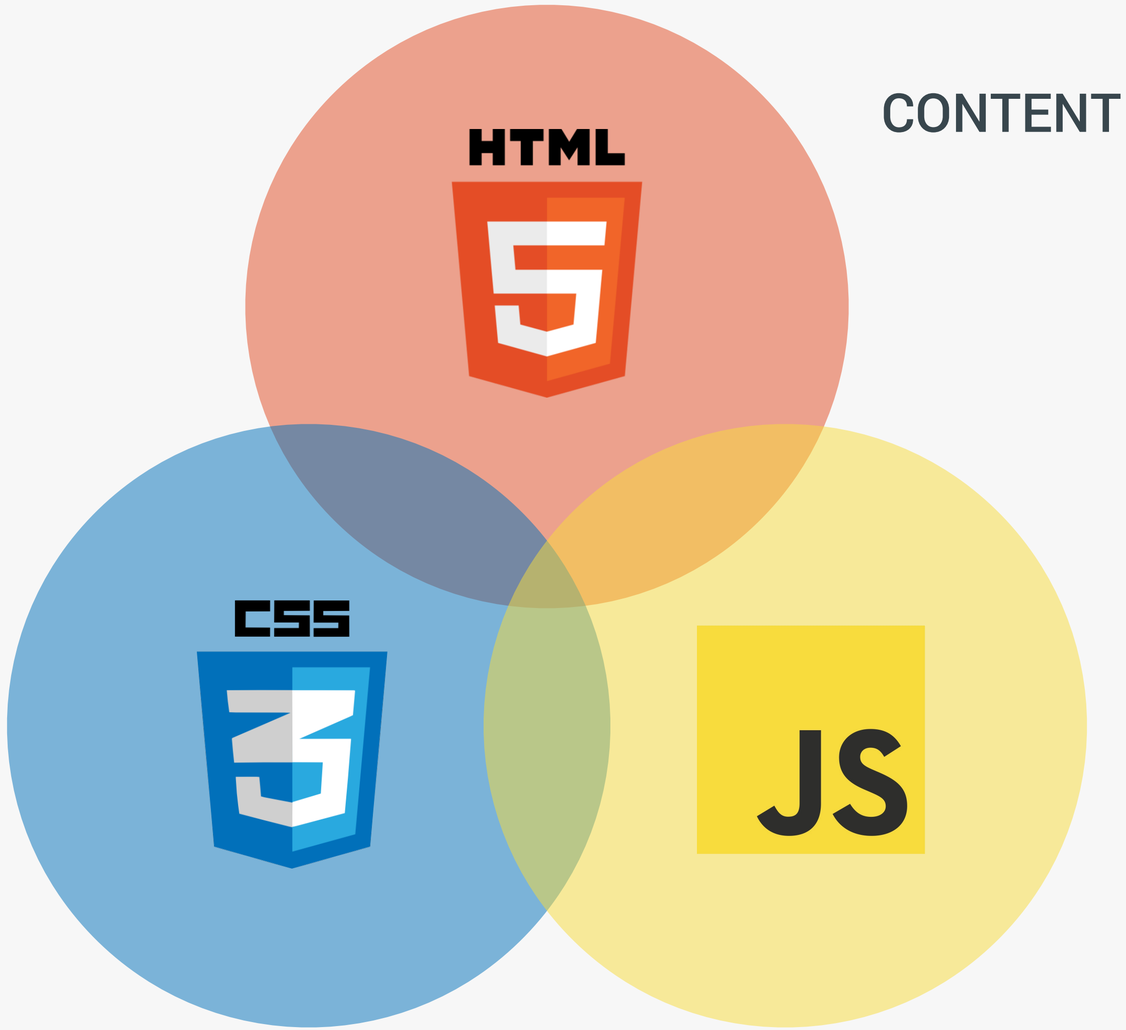
07 Next Generation JavaScript: Intro to ES6 / ES2015

08 Asynchronous JavaScript: Promises, Async/Await and AJAX

09 Modern JavaScript: Using ES6, NPM, Babel and Webpack

SECTION 2 — JS LANGUAGE BASICS

THE ROLE OF JAVASCRIPT IN WEB DEVELOPMENT



CONTENT

PRESENTATION

DYNAMIC EFFECTS/
PROGRAMMING

NOUNS, ADJECTIVES AND VERBS

CONTENT

PRESENTATION

DYNAMIC EFFECTS/
PROGRAMMING

NOUNS

`<p></p>`

means “paragraph”

ADJECTIVES

```
p {color: red;}
```

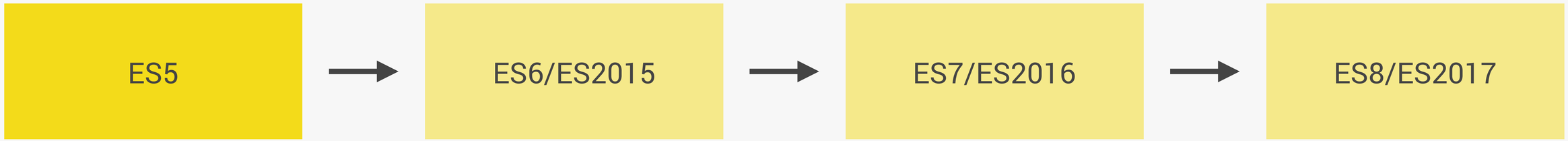
means “the paragraph text is red”

VERBS

```
p.hide();
```

means “hide the paragraph”

JAVASCRIPT VERSIONS... (MORE ABOUT THIS LATER)



PRIMITIVE JAVASCRIPT DATA TYPES

1. **Number:** Floating point numbers, for decimals and integers
2. **String:** Sequence of characters, used for text
3. **Boolean:** Logical data type that can only be `true` or `false`
4. **Undefined:** Data type of a variable that does not have a value yet
5. **Null:** Also means ‘non-existent’

JavaScript has dynamic typing: data types are **automatically** assigned to variables

BASIC BOOLEAN LOGIC: NOT, AND & OR

var A

var B	AND	TRUE	FALSE
	TRUE	TRUE	FALSE
	FALSE	FALSE	FALSE

var A

var B	OR	TRUE	FALSE
	TRUE	TRUE	TRUE
	FALSE	TRUE	FALSE

- `AND (&&)` => `true` if **ALL** are `true`
- `OR (||)` => `true` if **ONE** is `true`
- `NOT (!)` => inverts `true`/`false` value

```
var age = 16;

age >= 20;     // => false
age < 30;      // => true
!(age < 30);   // => false

age >= 20 && age < 30; // =>
age >= 20 || age < 30; // =>
```

A (VERY) SHORT HISTORY OF JAVASCRIPT

- **1996:** Changed from LiveScript to JavaScript to attract Java developers. **JavaScript has almost nothing to do with Java** ☝️
- **1997:** ES1 (ECMAScript 1) became the first version of the JavaScript language standard:
  - ECMAScript: The language standard;
  - JavaScript: The language in practice.
- **2009:** ES5 (ECMAScript 5) was released with lots of new features.
- **2015:** ES6/ES2015 (ECMAScript 2015) was released: **the biggest update to the language ever!**
- **2015:** Changed to an **annual release cycle** 🙏
- **2016/2017/2018/2019/…:** Release of ES2016/ES2017/ES2018/ES2019/…

JAVASCRIPT TODAY: WHICH VERSION TO USE?

ES5

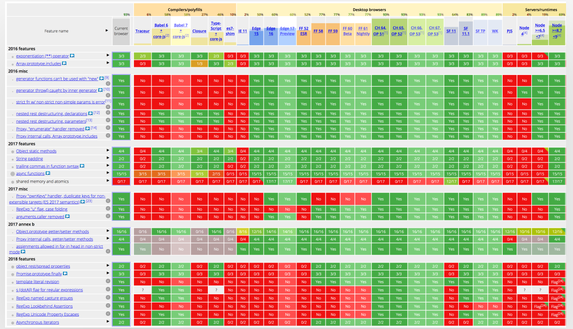
- Fully supported in all browsers;
- Ready to be used today 👍

ES6/ES2015

ES7/ES2016

ES8/ES2017

- Well supported in all **modern** browsers
- No support in older browsers;
- Can use **most** features in production with transpiling and polyfilling (converting to ES5) 😃



http://kangax.github.io/compat-table

ES9/ES2018

ES10/ES2019

- Future versions, together called ESNext;
- Some features supported in modern browsers;
- Can already use **some** features in production with transpiling and polyfilling 😊

(As of mid-2018)

WE USE ES5 AND ES6 IN THIS COURSE: WHY?

ES5

- JavaScript fundamentals
- How the language works
- DOM manipulation project
- Advanced language features
- Huge real project

ES6+

- ES6/ES2015 introduction
- Asynchronous JavaScript
- AJAX and API calls
- Modern dev setups (Webpack and Babel)
- Huge real project

- You will have to understand ES5 today and in the future;
- Many tutorials and code you find online today are still in ES5;
- When working on older codebases, these will be written in ES5;
- It’s better and easier to learn the fundamentals in ES5, and then update to ES6+.

SECTION 3 — HOW JAVASCRIPT WORKS BEHIND THE SCENES

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

HOW JAVASCRIPT WORKS BEHIND THE SCENES

LECTURE

HOW OUR CODE IS EXECUTED: JAVASCRIPT PARSERS AND ENGINES

WHAT HAPPENS TO OUR CODE?

OUR CODE

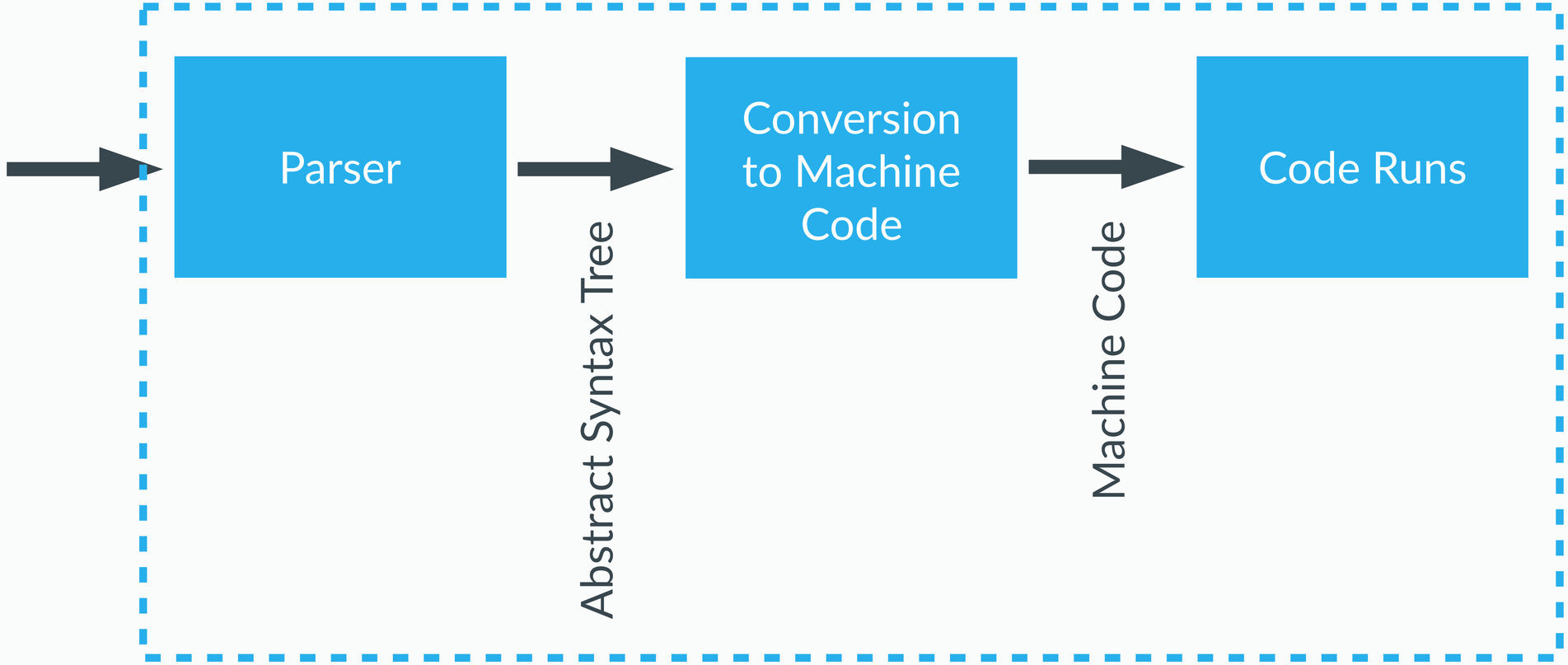
```
function calculateAge(yearOfBirth) {
    return 2016 - yearOfBirth;
}

var johnsAge = calculateAge(1990);

function yearsUntilRetirement(name, yearOfBirth) {
    var age = calculateAge(yearOfBirth);
    var retirement = 65 - age;
    if (retirement >= 0) {
        console.log(name + ' retires in ' + retirement + ' years.');
    } else {
        console.log(name + ' is already retired.');
    }
}

yearsUntilRetirement('John', 1990);
```

JAVASCRIPT ENGINE



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

HOW JAVASCRIPT WORKS BEHIND THE SCENES

LECTURE

EXECUTION CONTEXTS AND THE EXECUTION STACK

EXECUTION CONTEXTS

Execution Context

(A box, a container, or a wrapper which stores variables and in which a piece of our code is evaluated and executed)

```
function calculateAge(yearOfBirth) {
    return 2016 - yearOfBirth;
}

var johnsAge = calculateAge(1990);

function yearsUntilRetirement(name, yearOfBirth) {
    var age = calculateAge(yearOfBirth);
    var retirement = 65 - age;
    if (retirement >= 0) {
        console.log(name + ' retires in ' + retirement + ' years.');
    } else {
        console.log(name + ' is already retired.');
    }
}

yearsUntilRetirement('John', 1990);
```

THE DEFAULT

Global Execution Context

- Code that is **not inside any function**
- Associated with the **global object**
- In the browser, that's the `window` object

```
lastName === window.lastName
// true
```

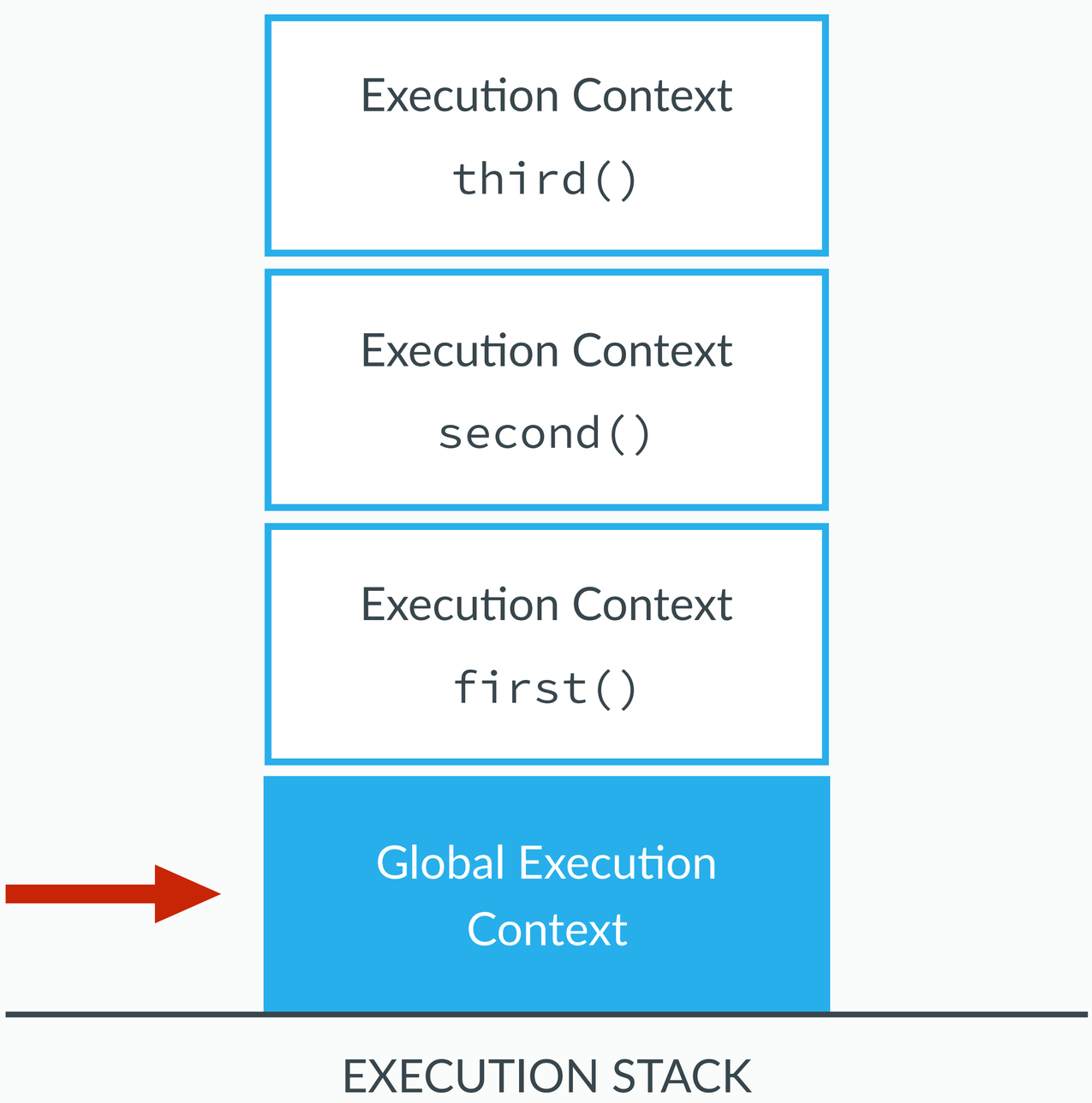
```
var name = 'John';

function first() {
    var a = 'Hello!';
    second();
    var x = a + name;
}

function second() {
    var b = 'Hi!';
    third();
    var z = b + name;
}

function third() {
    var c = 'Hey!';
    var z = c + name;
}

first();
```



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

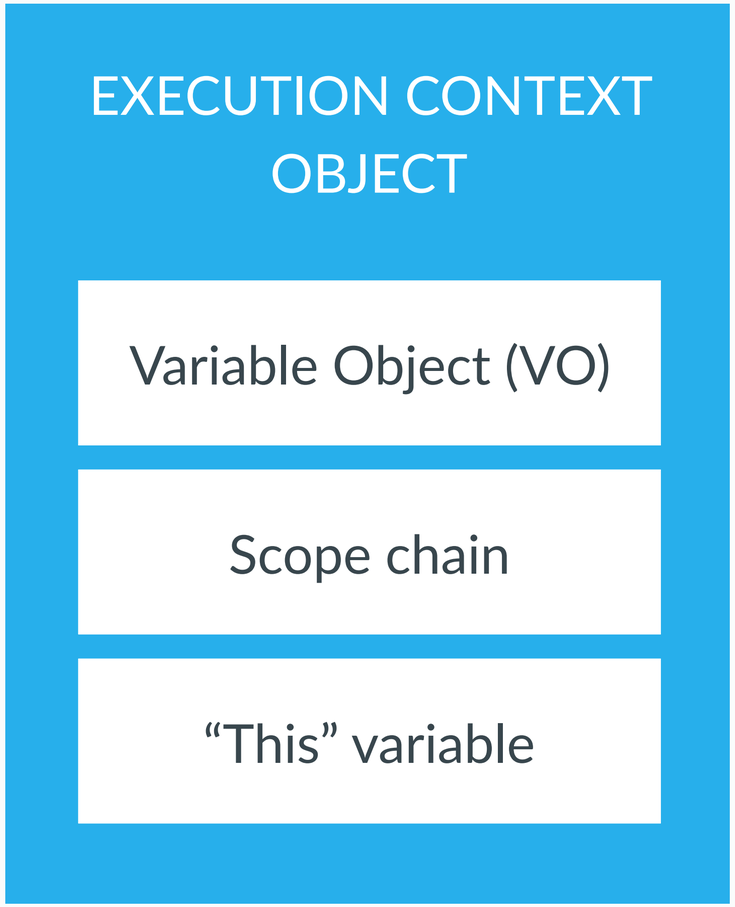
SECTION

HOW JAVASCRIPT WORKS BEHIND THE SCENES

LECTURE

EXECUTION CONTEXTS IN DETAIL: CREATION AND EXECUTION PHASES AND HOISTING

THE EXECUTION CONTEXT IN DETAIL



1. Creation phase

A) Creation of the Variable Object (VO)

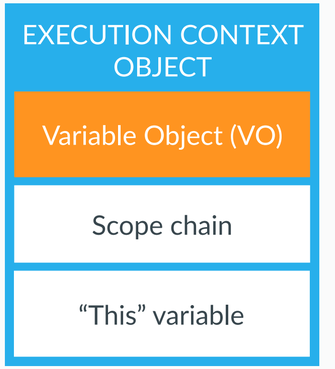
B) Creation of the scope chain

C) Determine value of the ‘this’ variable

2. Execution phase

The code of the function that generated the current execution context is ran line by line

THE VARIABLE OBJECT



- The `argument` object is created, containing all the arguments that were passed into the function.
- Code is scanned for **function declarations**: for each function, a property is created in the Variable Object, **pointing to the function**.
- Code is scanned for **variable declarations**: for each variable, a property is created in the Variable Object, and set to `undefined`.

HOISTING

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

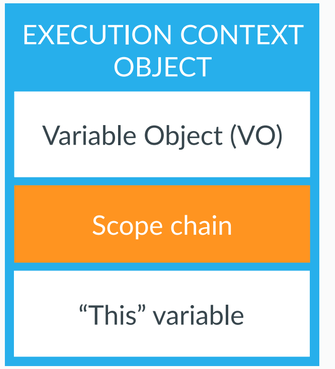
SECTION

HOW JAVASCRIPT WORKS BEHIND THE SCENES

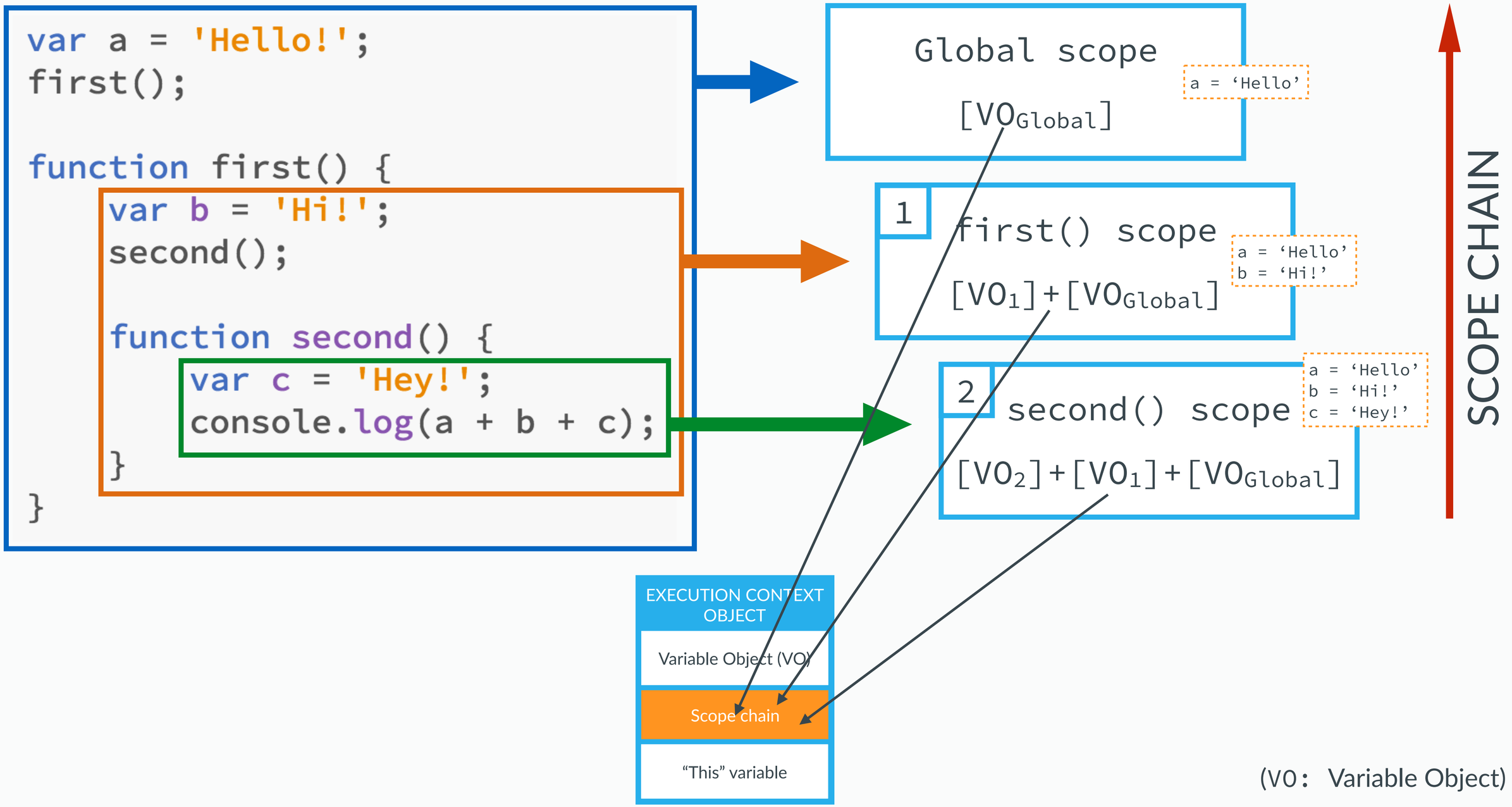
LECTURE

SCOPING AND THE SCOPE CHAIN

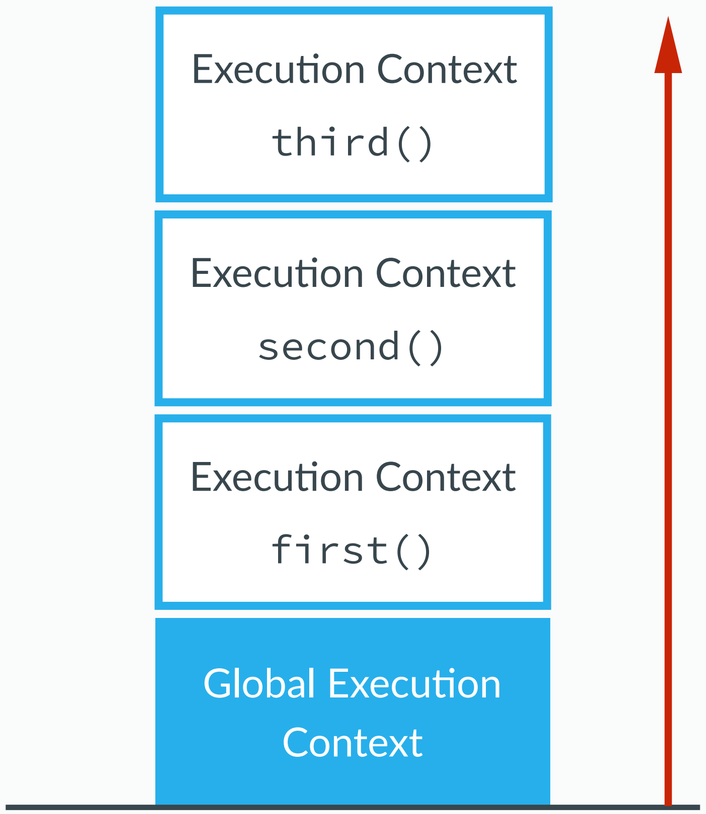
SCOPING IN JAVASCRIPT



- Scoping answers the question “where can we access a certain variable?”
- **Each new function creates a scope**: the space/environment, in which the variables it defines are accessible.
- **Lexical scoping**: a function that is lexically within another function gets access to the scope of the outer function.


var a = 'Hello!';
first();
function first() {
var b = 'Hi!';
second();
function second() {
var c = 'Hey!';
console.log(a + b + c);
}
}
Global scope
[VOGlobal]
a = ‘Hello’
1
first() scope
[VO1]+[VOGlobal]
a = ‘Hello’
b = ‘Hi!’
2
second() scope
[VO2]+[VO1]+[VOGlobal]
a = ‘Hello’
b = ‘Hi!’
c = ‘Hey!’
SCOPE CHAIN
EXECUTION CONTEXT OBJECT
Variable Object (VO)
Scope chain
“This” variable
(VO: Variable Object)

EXECUTION STACK VS SCOPE CHAIN

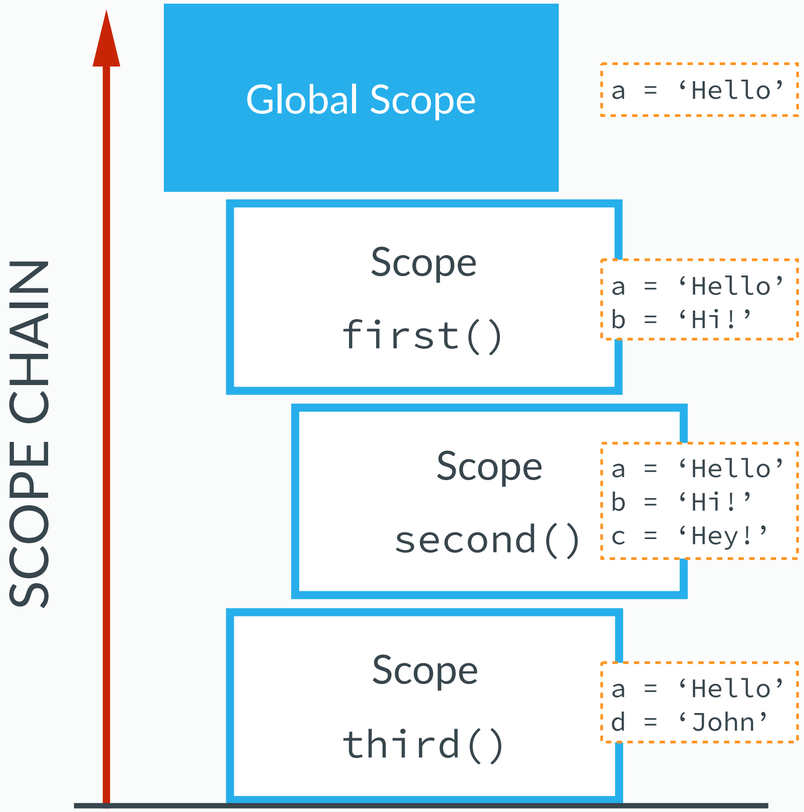



```
var a = 'Hello!';
first();

function first() {
    var b = 'Hi!';
    second();

    function second() {
        var c = 'Hey!';
        third()
    }
}

function third() {
    var d = 'John';
    console.log(a + b + c + d);
}
```



Order in which functions are **called**

≠

Order in which functions are **written lexically**

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

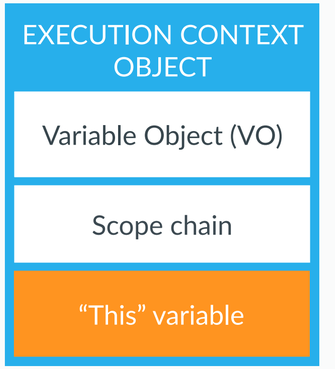
SECTION

HOW JAVASCRIPT WORKS BEHIND THE SCENES

LECTURE

THE 'THIS' KEYWORD

THE ‘THIS’ VARIABLE



- **Regular function call**: the `this` keyword points at the global object, (the `window` object, in the browser).
- **Method call**: the `this` variable points to the object that is calling the method.

- *The `this` keyword is not assigned a value until a function where it is defined is actually called.*

SECTION 4 — JAVASCRIPT IN THE BROWSER: DOM MANIPULATION AND EVENTS

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

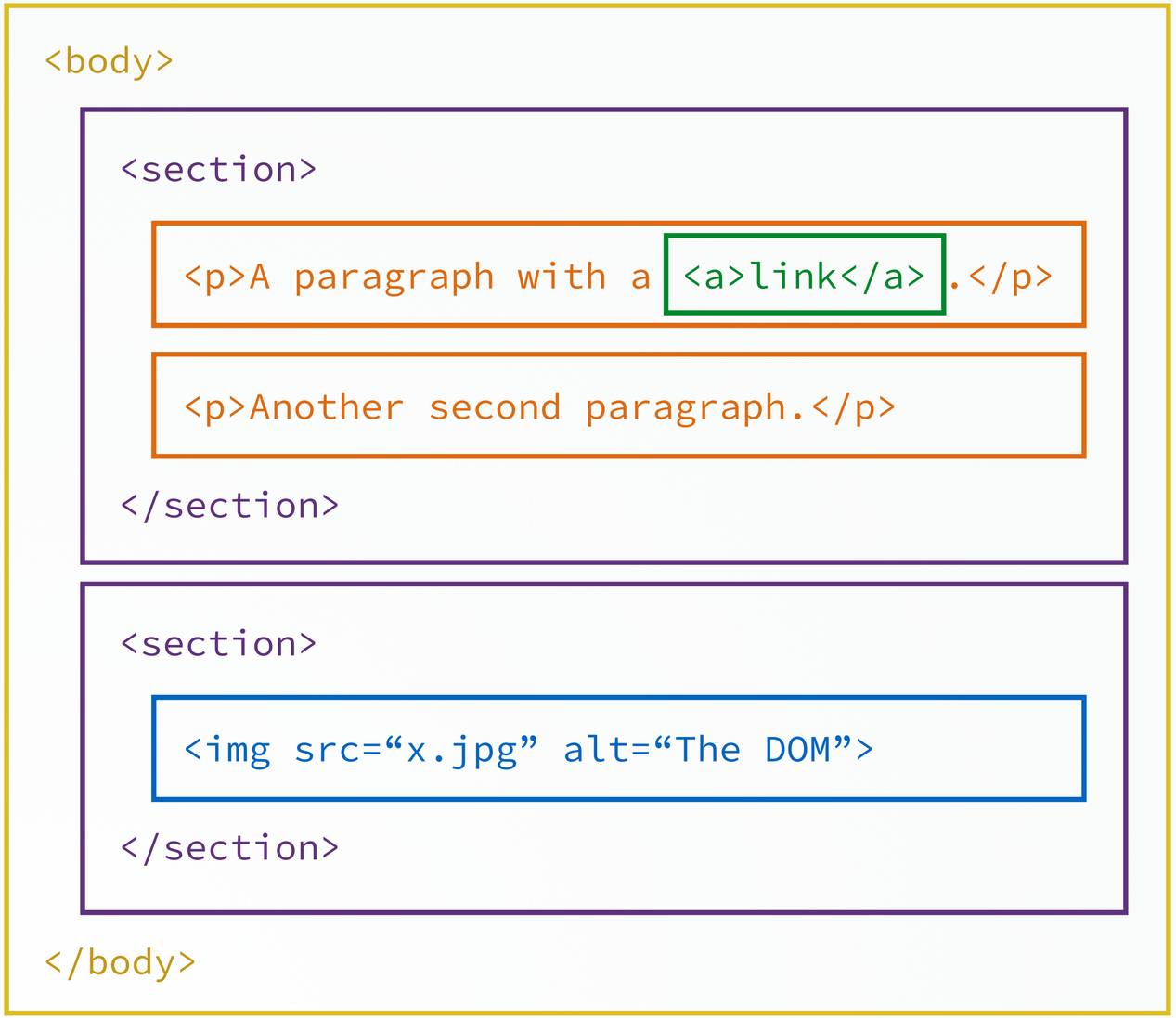
JAVASCRIPT IN THE BROWSER: DOM MANIPULATION AND EVENTS

LECTURE

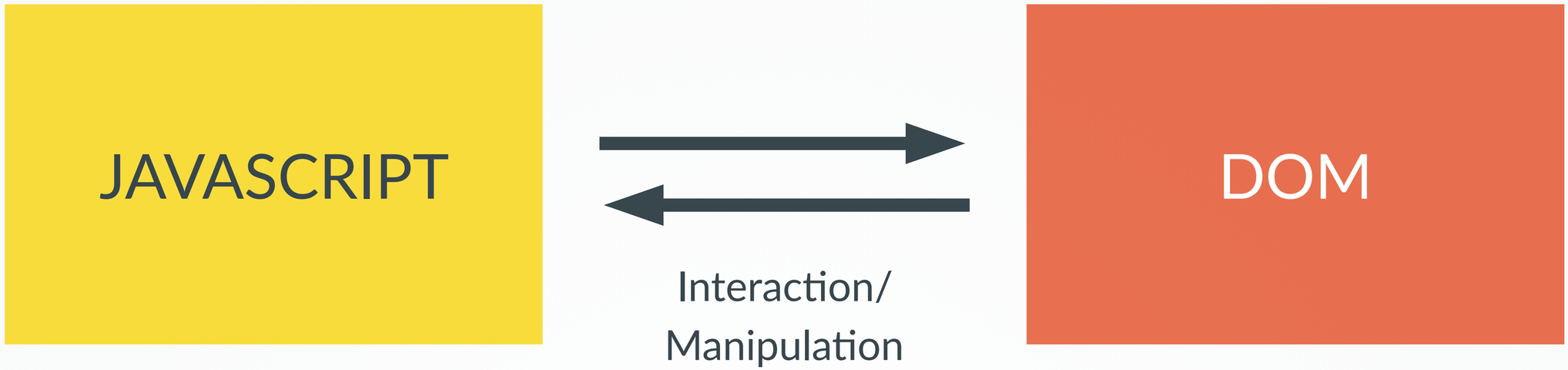
THE DOM AND DOM MANIPULATION

THE DOCUMENT OBJECT MODEL

- **DOM:** Document Object Model;
- Structured representation of an HTML document;
- The DOM is used to connect webpages to scripts like JavaScript;
- For each HTML box, there is an object in the DOM that we can access and interact with.

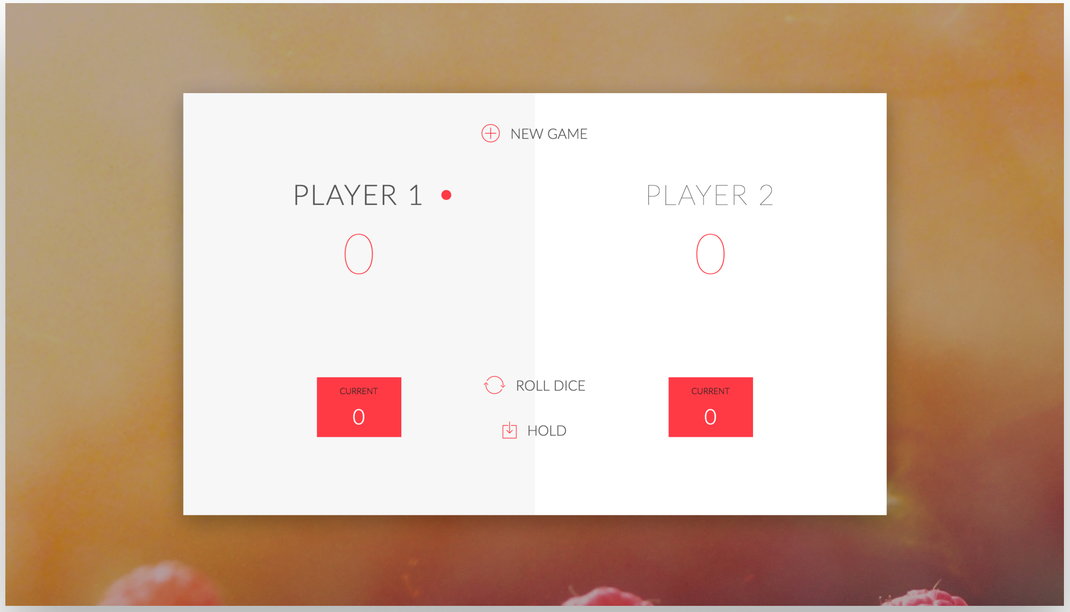


DOM MANIPULATION



WHAT YOU WILL LEARN IN THIS LECTURE

- How to create our fundamental game variables;
- How to generate a random number;
- How to manipulate the DOM;
- How to read from the DOM;
- How to change CSS styles.



JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

JAVASCRIPT IN THE BROWSER: DOM MANIPULATION AND EVENTS

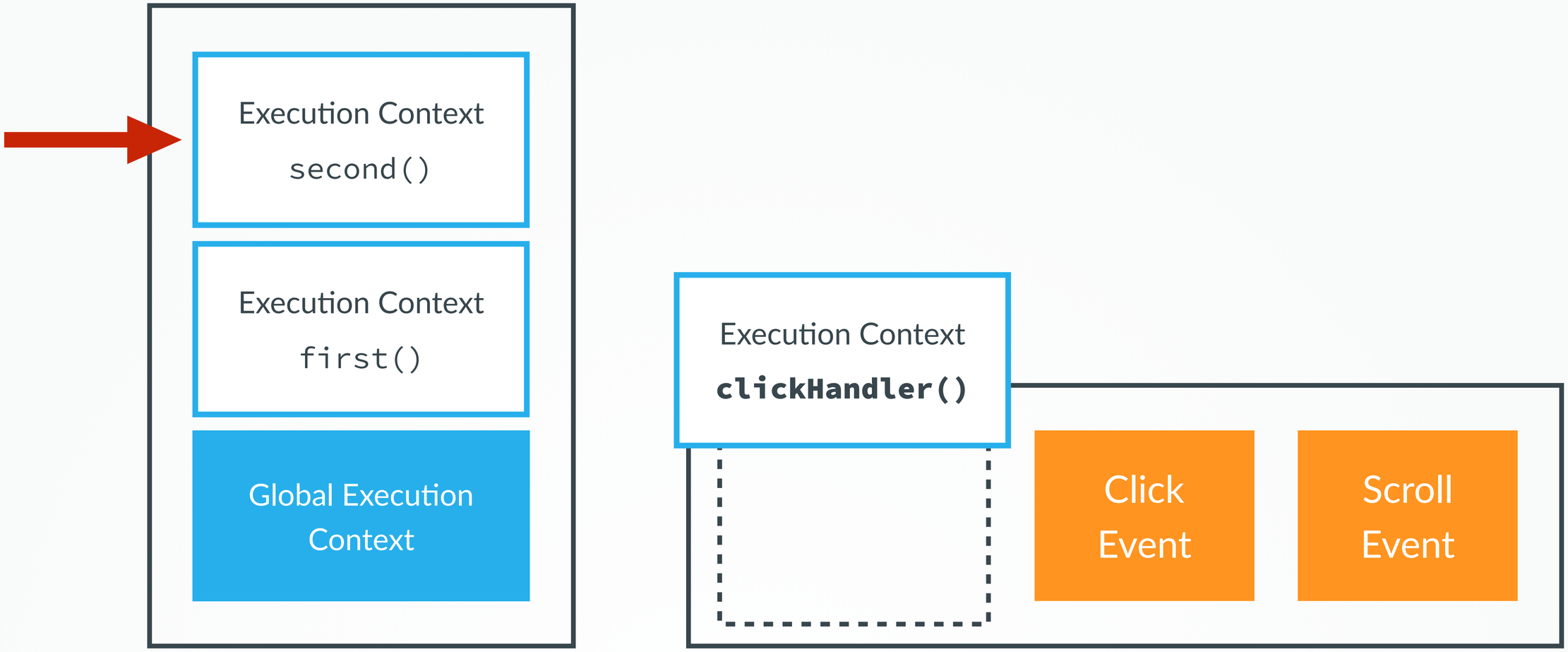
LECTURE

EVENTS AND EVENT HANDLING: ROLLING THE DICE

WHAT ARE EVENTS?

- **Events:** Notifications that are sent to notify the code that something happened on the webpage;
- Examples: clicking a button, resizing a window, scrolling down or pressing a key;
- **Event listener:** A function that performs an action based on a certain event. It waits for a specific event to happen.

HOW EVENTS ARE PROCESSED



SECTION 5 — ADVANCED JAVASCRIPT: OBJECTS AND FUNCTIONS

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

ADVANCED JAVASCRIPT: OBJECTS AND FUNCTIONS

LECTURE

EVERYTHING IS AN OBJECT: INHERITANCE AND THE PROTOTYPE CHAIN

OBJECTS IN JAVASCRIPT

Everything is an object.

(Well, almost everything)

PRIMITIVES

- Numbers
- Strings
- Booleans
- Undefined
- Null

EVERYTHING ELSE ...

- Arrays
- Functions
- Objects
- Dates
- Wrappers for Numbers, Strings, Booleans

... IS AN OBJECT

THE OBJECT ORIENTED PARADIGM

OBJECT-ORIENTED PROGRAMMING

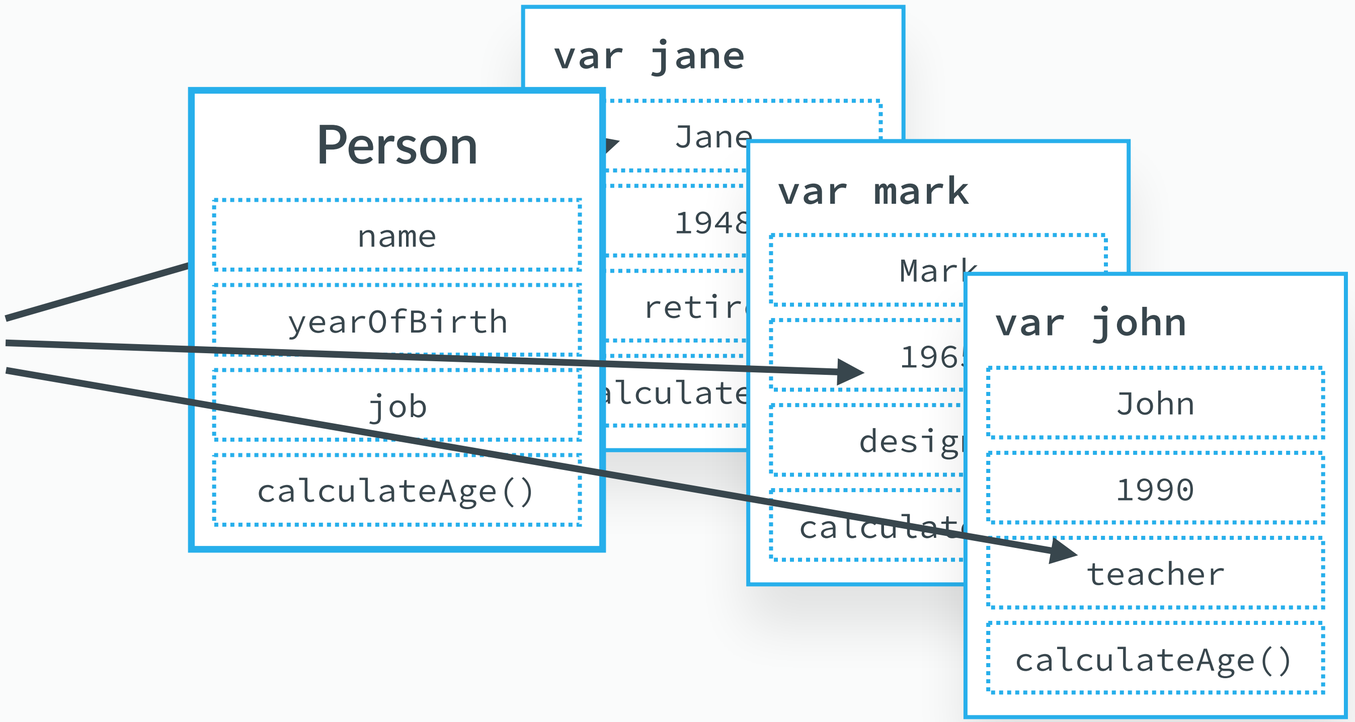
- Objects interacting with one another through methods and properties;
- Used to store data, structure applications into modules and keeping code clean.

```
var john = {
    name: 'John',
    yearOfBirth: 1990,
    isMarried: false
};
```

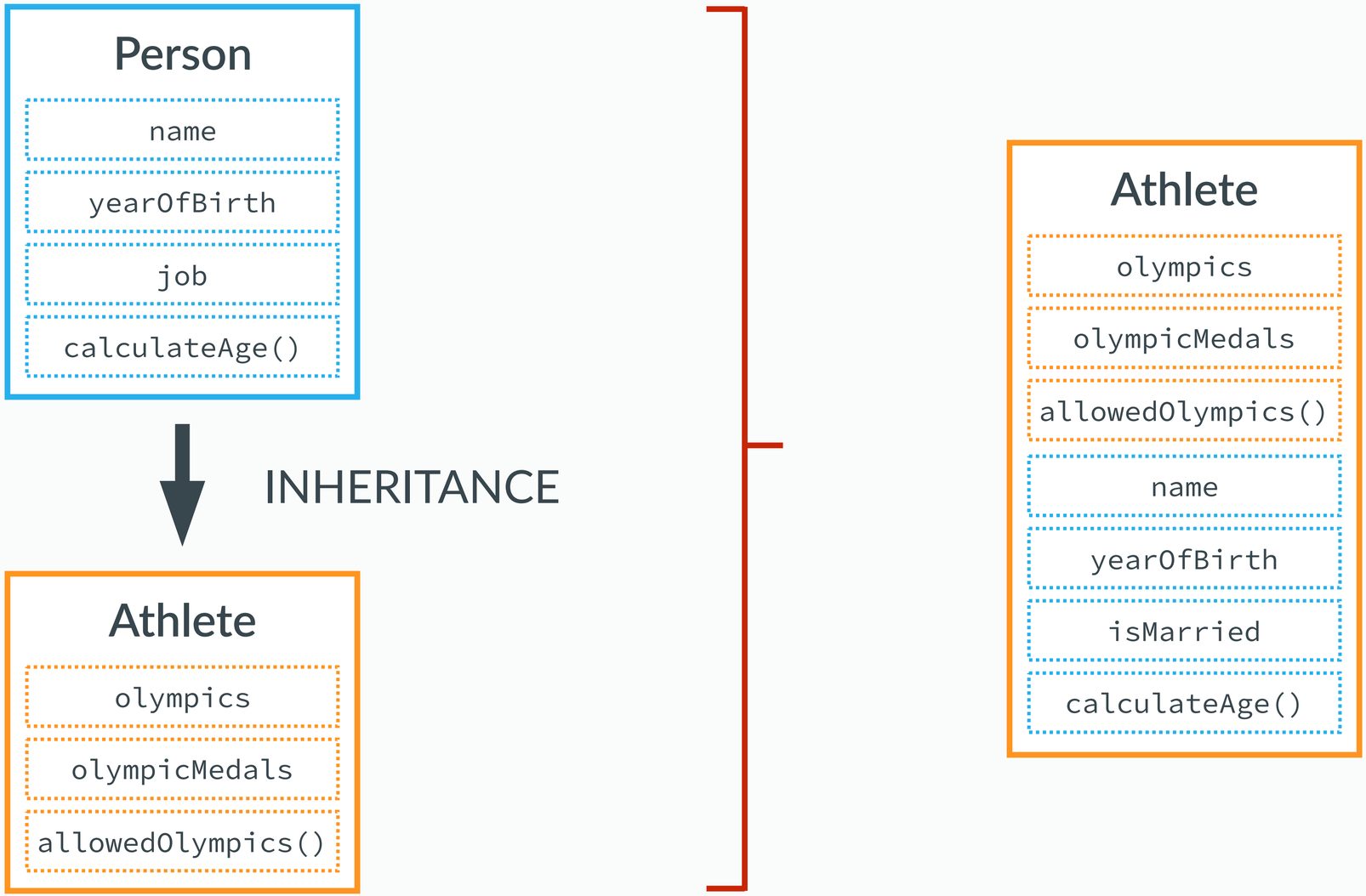
```
var jane = {
    name: 'Jane',
    yearOfBirth: 1969,
    isMarried: true
};
```

```
var mark = {
    name: 'Mark',
    yearOfBirth: 1948,
    isMarried: true
};
```

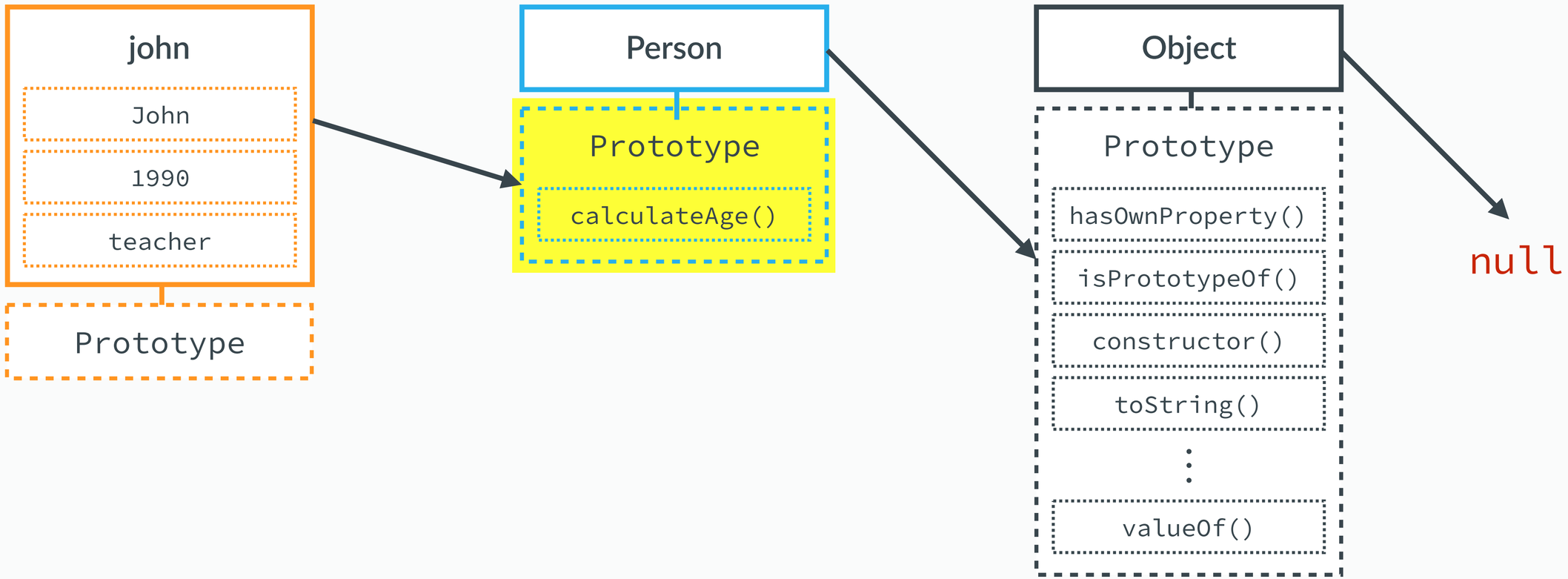
CONSTRUCTORS AND INSTANCES IN JAVASCRIPT



INHERITANCE IN GENERAL



INHERITANCE IN JAVASCRIPT: PROTOTYPES AND PROTOTYPE CHAINS



SUMMARY

- Every JavaScript object has a **prototype property**, which makes inheritance possible in JavaScript;
- The prototype property of an object is where we put methods and properties that we want **other objects to inherit**;
- The Constructor's prototype property is **NOT** the prototype of the Constructor itself, it's the prototype of **ALL** instances that are created through it;
- When a certain method (or property) is called, the search starts in the object itself, and if it cannot be found, the search moves on to the object's prototype. This continues until the method is found: **prototype chain**.

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

ADVANCED JAVASCRIPT: OBJECTS AND FUNCTIONS

LECTURE

CREATING OBJECTS: FUNCTION CONSTRUCTORS

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

ADVANCED JAVASCRIPT: OBJECTS AND FUNCTIONS

LECTURE

THE PROTOTYPE CHAIN IN THE CONSOLE

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

ADVANCED JAVASCRIPT: OBJECTS AND FUNCTIONS

LECTURE

CREATING OBJECTS: OBJECT.CREATE

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

ADVANCED JAVASCRIPT: OBJECTS AND FUNCTIONS

LECTURE

PRIMITIVES VS. OBJECTS

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

ADVANCED JAVASCRIPT: OBJECTS AND FUNCTIONS

LECTURE

FIRST CLASS FUNCTIONS: PASSING FUNCTIONS AS ARGUMENTS

FUNCTIONS ARE ALSO OBJECTS IN JAVASCRIPT

- A function is an instance of the Object type;
- A function behaves like any other object;
- We can store functions in a variable;
- We can pass a function as an argument to another function;
- We can return a function from a function.

FIRST-CLASS FUNCTIONS

CLOSURES

CLOSURES SUMMARY

An inner function has always access to the variables and parameters of its outer function, even after the outer function has returned.

HOW CLOSURES WORK

CLOSURES

CLOSURES SUMMARY (AGAIN :)

An inner function has always access to the variables and parameters of its outer function, even after the outer function has returned.

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

ADVANCED JAVASCRIPT: OBJECTS AND FUNCTIONS

LECTURE

BIND, CALL AND APPLY

NOT SO FAST...

(LET'S THINK TOGETHER)

STRUCTURING OUR CODE WITH MODULES

UI MODULE

DATA MODULE

CONTROLLER MODULE

TO-DO LIST

- Add event handler
- Get input values
- Add the new item to our data structure
- Add the new item to the UI
- Calculate budget
- Update the UI

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

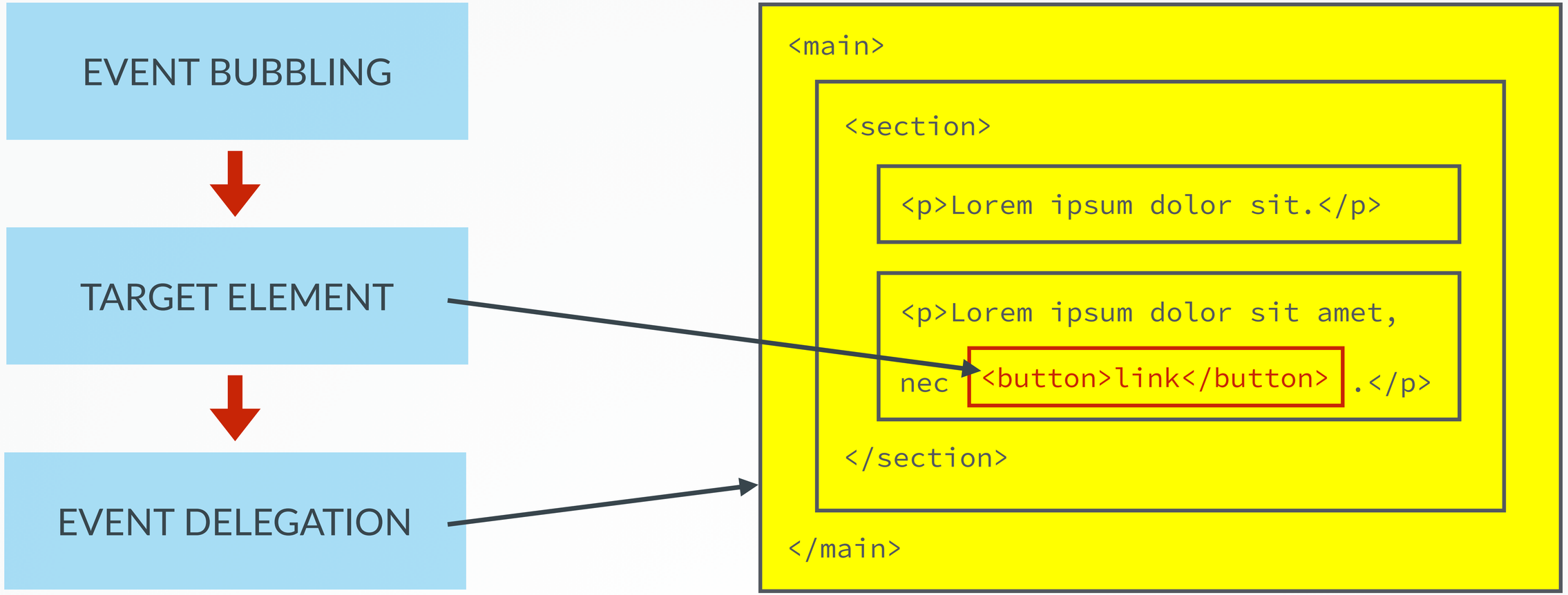
SECTION

PUTTING IT ALL TOGETHER: THE BUDGET APP PROJECT

LECTURE

IMPLEMENTING THE MODULE PATTERN

EVENT BUBBLING, TARGET ELEMENT AND EVENT DELEGATION



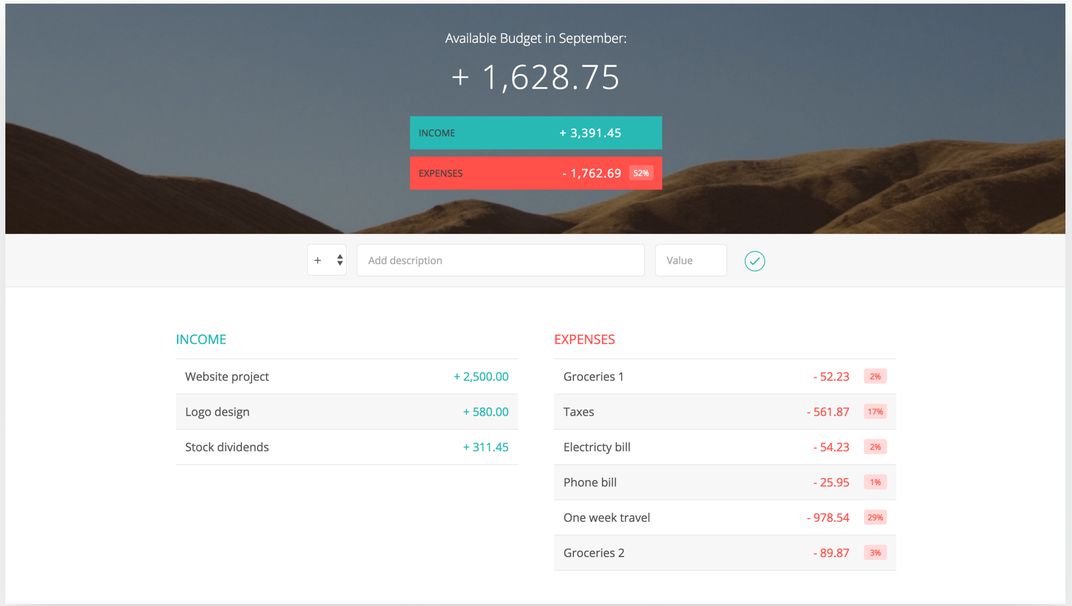
WHEN TO USE EVENT DELEGATION

USE CASES FOR EVENT DELEGATION

1. When we have an element with lots of child elements that we are interested in;
2. When we want an event handler attached to an element that is not yet in the DOM when our page is loaded.

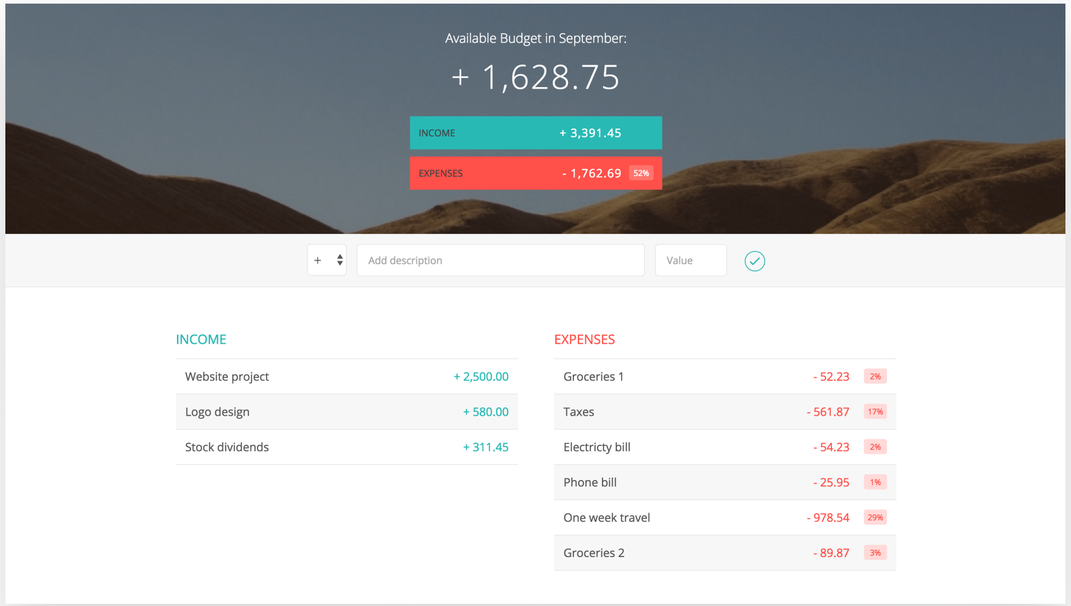
WHAT YOU WILL LEARN IN THIS LECTURE

- How to use event delegation in practice;
- How to use IDs in HTML to connect the UI with the data model;
- How to use the `parentNode` property for DOM traversing.

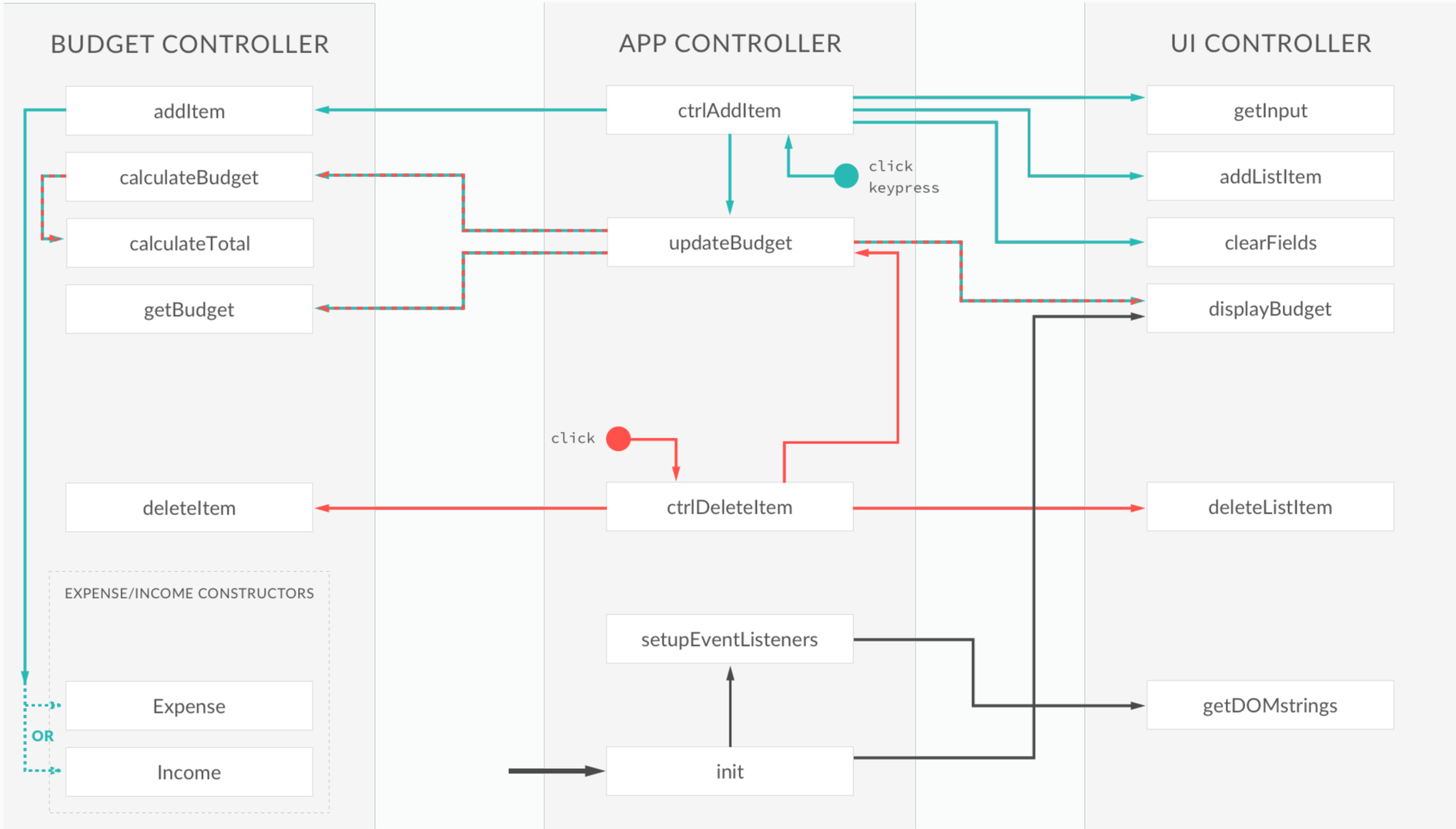


WHAT YOU WILL LEARN IN THIS LECTURE

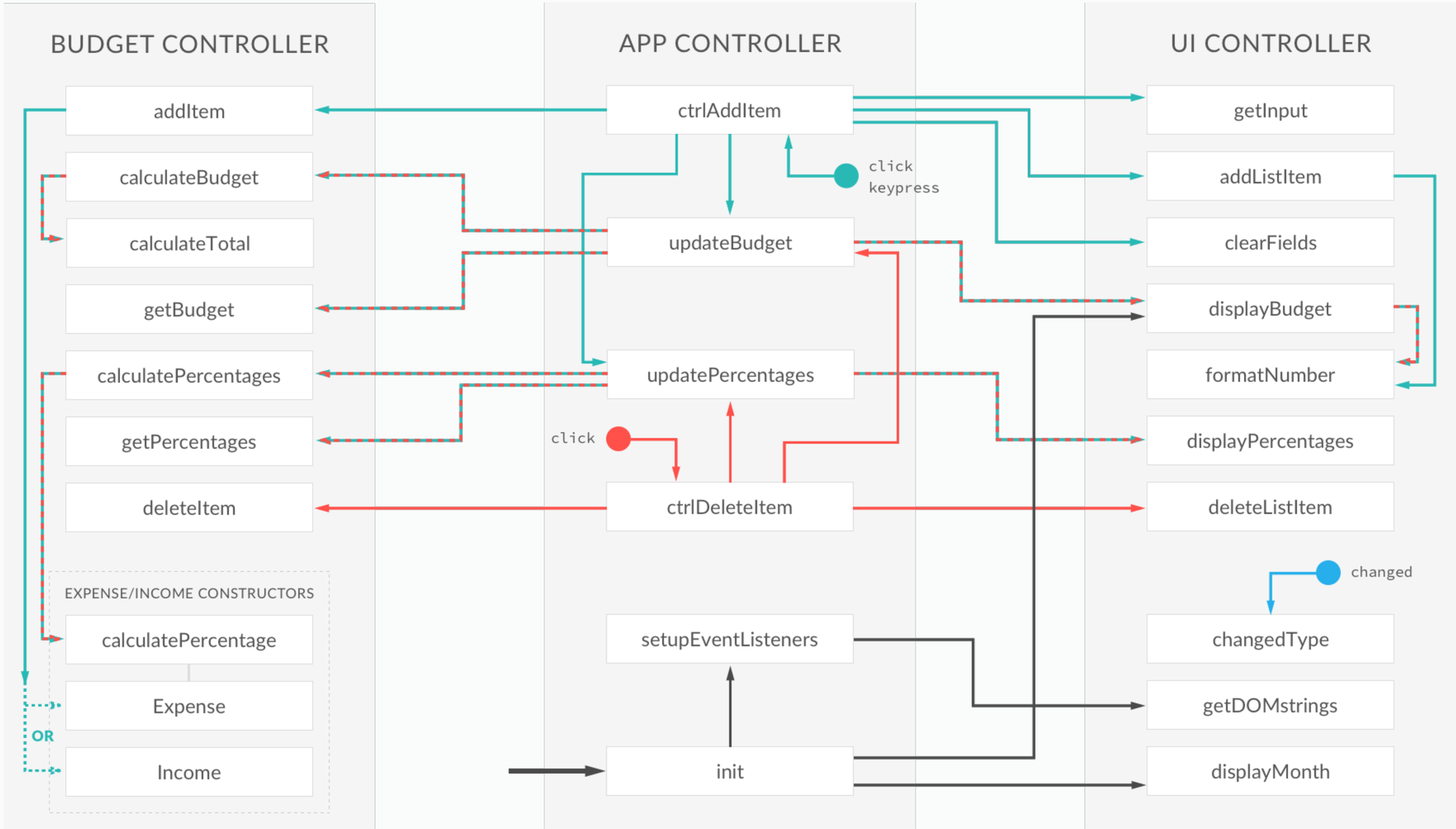
- More DOM manipulation: how to remove an element from the DOM.



AFTER STEP 2... (THAT WE JUST COMPLETED)



OUR FINAL ARCHITECTURE



SECTION 7 — GET READY FOR THE FUTURE: ECMASCRIPT2015

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

NEXT GENERATION JAVASCRIPT: INTRO TO ES6 / ES2015

LECTURE

SECTION INTRO

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

NEXT GENERATION JAVASCRIPT: INTRO TO ES6 / ES2015

LECTURE

WHAT'S NEW IN ES6 / ES2015

JAVASCRIPT VERSIONS: QUICK RECAP

ES6/ES2015

ES7/ES2016

ES8/ES2017

- Well supported in all **modern** browsers
- No support in older browsers;
- Can use **most** features in production with transpiling and polyfilling (converting to ES5) 😃

NEW ES6 FEATURES WE'LL COVER IN THIS SECTION

- Variable Declarations with `let` and `const`
- Blocks and IIFEs
- Strings
- Arrow Functions
- Destructuring
- Arrays
- The Spread Operator
- Rest and Default Parameters
- Maps
- Classes and subclasses

LATER...

- Promises
- Native modules

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

NEXT GENERATION JAVASCRIPT: INTRO TO ES6 / ES2015

LECTURE

VARIABLE DECLARATIONS WITH LET AND CONST

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

NEXT GENERATION JAVASCRIPT: INTRO TO ES6 / ES2015

LECTURE

BLOCKS AND IIFES

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

NEXT GENERATION JAVASCRIPT: INTRO TO ES6 / ES2015

LECTURE

STRINGS IN ES6 / ES2015

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

NEXT GENERATION JAVASCRIPT: INTRO TO ES6 / ES2015

LECTURE

ARROW FUNCTIONS: BASICS

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

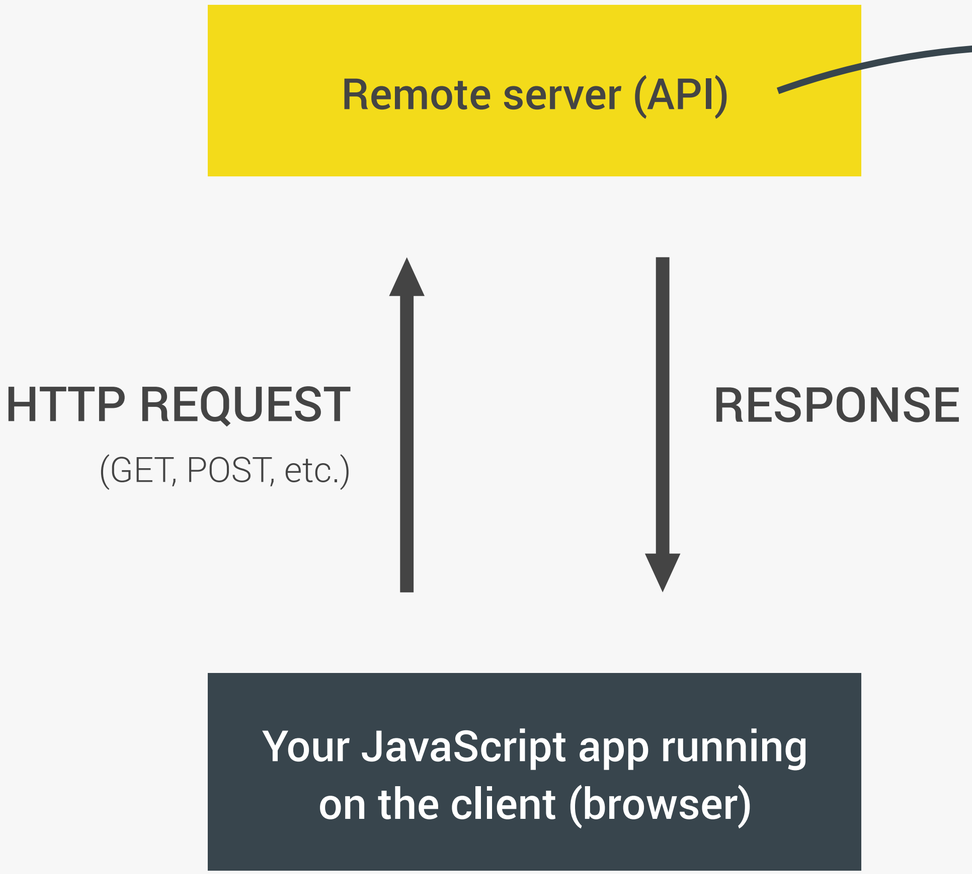
NEXT GENERATION JAVASCRIPT: INTRO TO ES6 / ES2015

LECTURE

ARROW FUNCTIONS: LEXICAL 'THIS' KEYWORD

WHAT ARE AJAX AND APIS?

ASYNCHRONOUS **J**AVASCRIPT **A**ND **X**ML



APPLICATION **P**ROGRAMMING **I**NTERFACE

- Your own API, for data coming from your own server
- 3rd-party APIs:
  - Google Maps
  - Embed Youtube videos
  - Weather data
  - Movies data
  - Send email or SMS
  - Thousands of possibilities...

SECTION 9 — MODERN JAVASCRIPT

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

MODERN JAVASCRIPT: USING ES6, NPM, BABEL AND WEBPACK

LECTURE

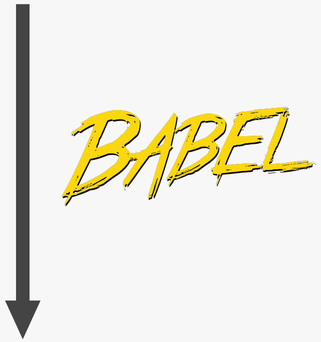
SECTION INTRO

MODERN JAVASCRIPT: A BRIEF OVERVIEW

3rd-party packages

— Libraries/frameworks
— Development tools

ES6/ESNext



ES5

ES6 Modules

Bundle

Putting it all together with an automated development setup powered by npm scripts

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

BUILD REAL-WORLD PROJECTS!

@JONASSCHMEDTMAN

SECTION

MODERN JAVASCRIPT: USING ES6, NPM, BABEL AND WEBPACK

LECTURE

A MODERN SETUP: CONFIGURING WEBPACK

OUR MODEL-VIEW-CONTROLLER ARCHITECTURE (MVC)