JAVASCRIPT FUNDAMENTALS – PART 1

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

@JONASSCHMEDTMAN

SECTION

JAVASCRIPT FUNDAMENTALS - PART 1

LECTURE

A BRIEF INTRODUCTION TO JAVASCRIPT

JS

WHAT IS JAVASCRIPT?

JAVASCRIPT

We don't have to worry about complex stuff like memory management

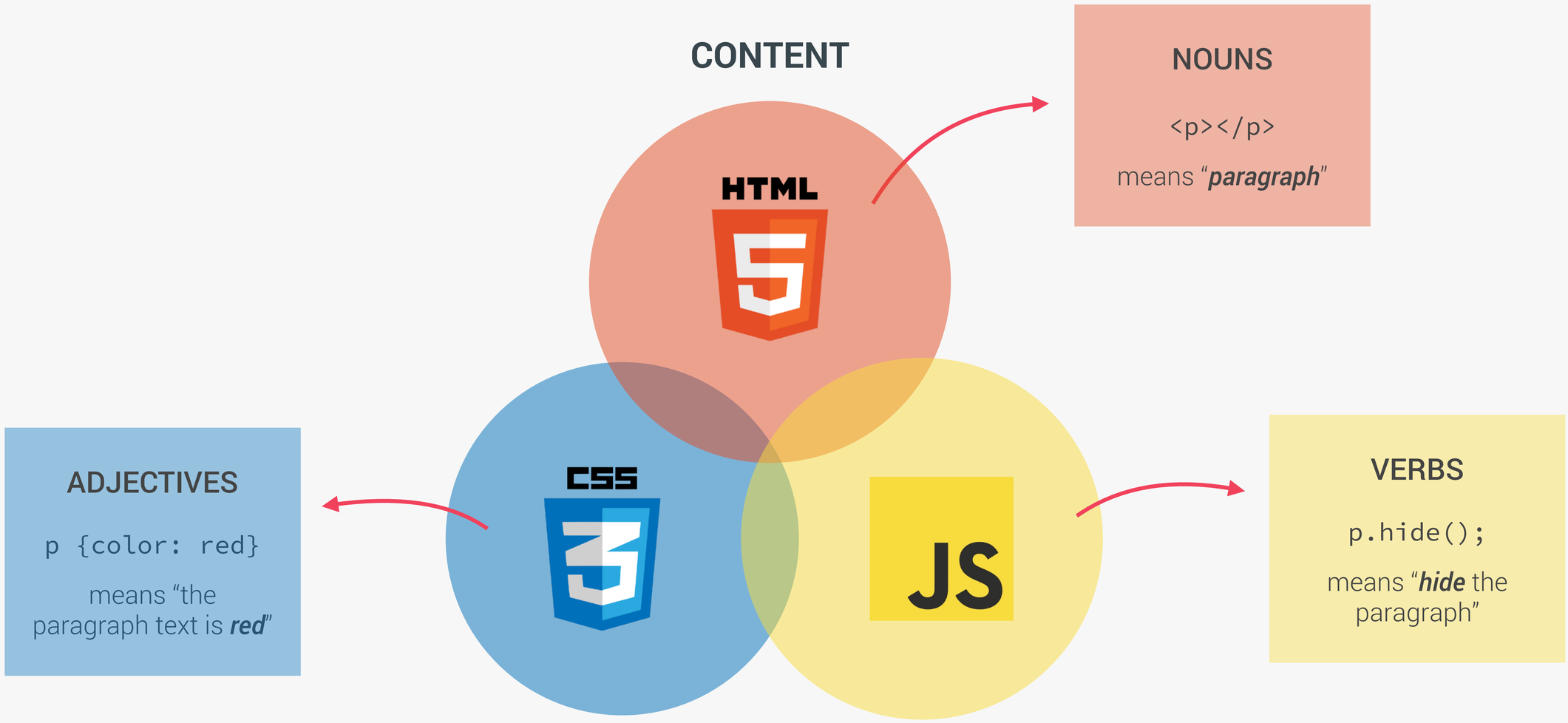
JAVASCRIPT IS A HIGH-LEVEL, OBJECT-ORIENTED, MULTI-PARADIGM PROGRAMMING LANGUAGE. 🤯

Based on objects, for storing most kinds of data

We can use different styles of programming

Instruct computer to *do* things

THE ROLE OF JAVASCRIPT IN WEB DEVELOPMENT



THERE IS NOTHING YOU CAN'T DO WITH JAVASCRIPT (WELL, ALMOST...)

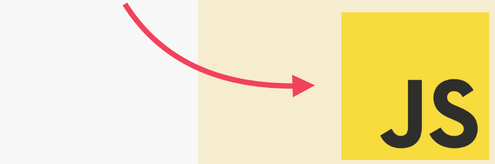
Front-end apps

Back-end apps

THIS COURSE

Dynamic effects and web applications in the browser 😍

Web applications on web servers

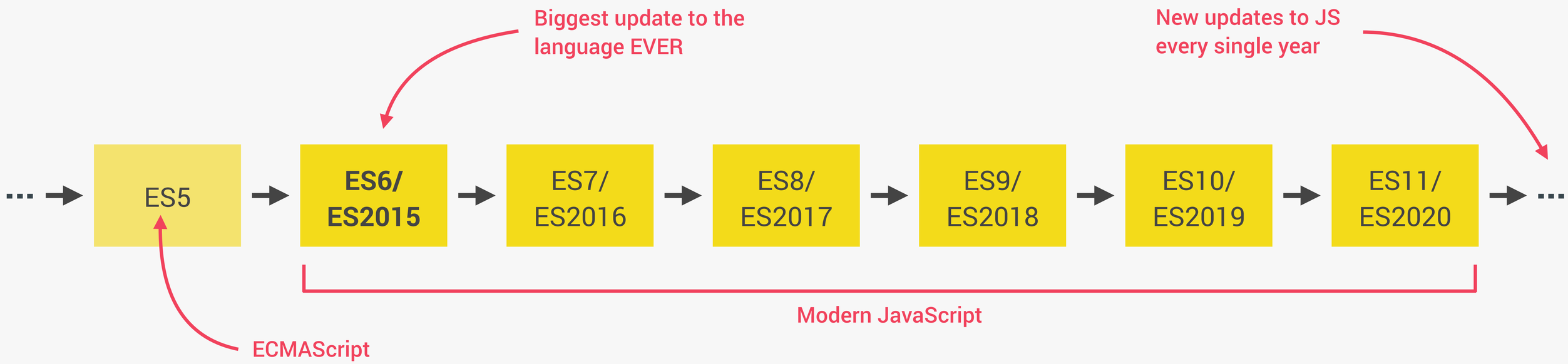


100% based on JavaScript. They might go away, **but JavaScript won't!**

Native mobile applications

Native desktop applications

JAVASCRIPT RELEASES... (MORE ABOUT THIS LATER)



🔥 Learn **modern JavaScript from the beginning**, but without forgetting the older parts!

🥳 **Let's finally get started!**

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

@JONASSCHMEDTMAN

SECTION

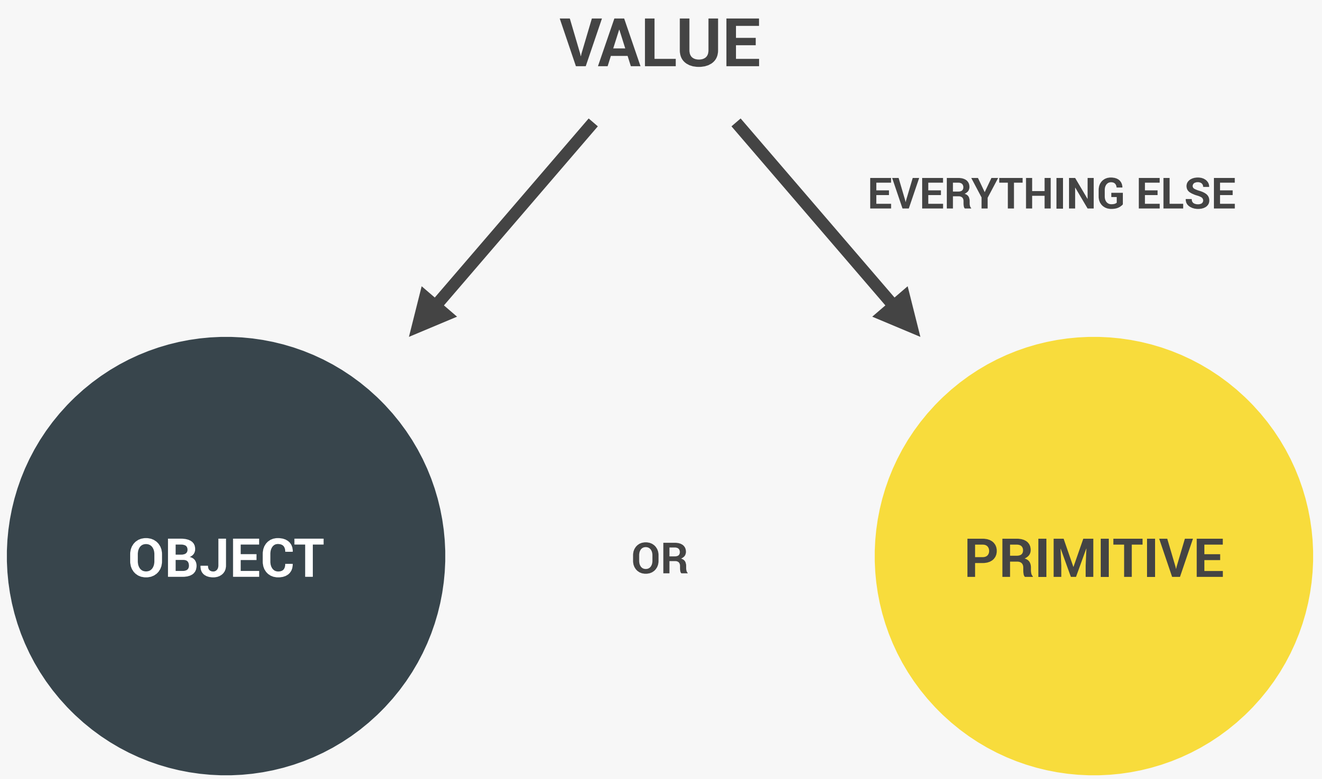
JAVASCRIPT FUNDAMENTALS - PART 1

LECTURE

DATA TYPES

JS

OBJECTS AND PRIMITIVES




```
let me = {
  name: 'Jonas'
};
```

```
let firstName = 'Jonas';
let age = 30;
```

THE 7 PRIMITIVE DATA TYPES

1. **Number:** Floating point numbers 👉 Used for decimals and integers
```
let age = 23;
```
2. **String:** Sequence of characters 👉 Used for text
```
let firstName = 'Jonas';
```
3. **Boolean:** Logical type that can only be `true` or `false` 👉 Used for taking decisions
```
let fullAge = true;
```
4. **Undefined:** Value taken by a variable that is not yet defined ('empty value')
```
let children;
```
5. **Null:** Also means 'empty value'
6. **Symbol (ES2015):** Value that is unique and cannot be changed *[Not useful for now]*
7. **BigInt (ES2020):** Larger integers than the Number type can hold

☝️ **JavaScript has dynamic typing:** We do ***not*** have to manually define the data type of the value stored in a variable. Instead, data types are determined **automatically**.

Value has type, NOT variable!

JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

@JONASSCHMEDTMAN

SECTION

JAVASCRIPT FUNDAMENTALS - PART 1

LECTURE

BOOLEAN LOGIC

JS

BASIC BOOLEAN LOGIC: THE AND, OR & NOT OPERATORS

A AND B

*"Sarah has a driver's license **AND** good vision"*

B \ A	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

Possible values

Results of operation, depending on 2 variables

👇

`true` when **ALL** are `true`

No matter how many variables

A OR B

*"Sarah has a driver's license **OR** good vision"*

B \ A	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

👇

`true` when **ONE** is `true`

NOT A, NOT B

👇

Inverts `true`/`false` value

👉 **EXAMPLE:**

A: Sarah has a driver's license

B: Sarah has good vision

Boolean variables that can be either TRUE or FALSE

AN EXAMPLE 👩‍💻

```
age = 16
```

BOOLEAN VARIABLES

👉 **A**: Age is greater or equal 20 → `false`

👉 **B**: Age is less than 30 → `true`

LET'S USE OPERATORS!

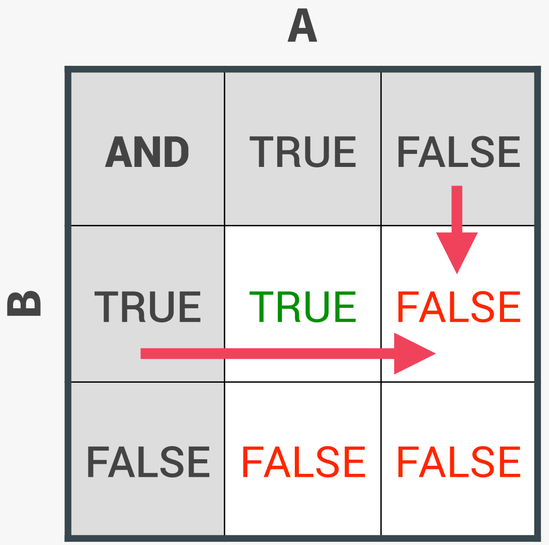
👉 `!A` → `true`
`false`

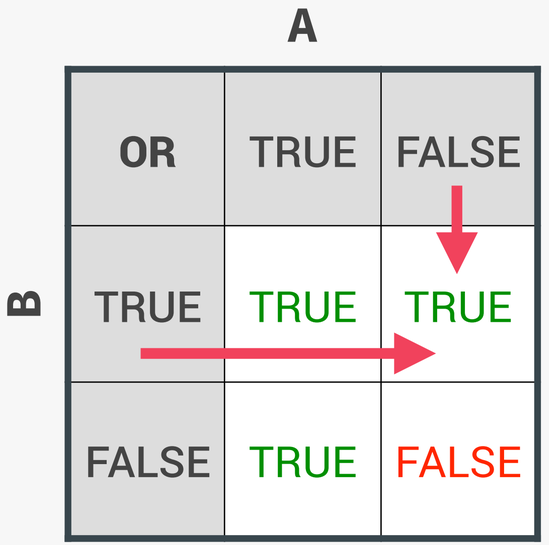
👉 `A AND B` → `false`
`false` `true`

👉 `A OR B` → `true`
`false` `true`

👉 `!A AND B` → `true`
`true` `true`

👉 `A OR !B` → `false`
`false` `false`





JONAS.IO
SCHMEDTMANN

THE COMPLETE JAVASCRIPT COURSE

FROM ZERO TO EXPERT!

@JONASSCHMEDTMAN

SECTION

JAVASCRIPT FUNDAMENTALS - PART 1

LECTURE

JAVASCRIPT RELEASES: ES5, ES6+ AND ESNEXT

JS

A BRIEF HISTORY OF JAVASCRIPT

1995

- 👉 Brendan Eich creates the **very first version of JavaScript in just 10 days**. It was called Mocha, but already had many fundamental features of modern JavaScript!

1996

- 👉 Mocha changes to LiveScript and then to JavaScript, in order to attract Java developers. However, **JavaScript has almost nothing to do with Java** ☝️
- 👉 Microsoft launches IE, **copying JavaScript from Netscape** and calling it JScript;

1997

- 👉 With a need to standardize the language, ECMA releases ECMAScript 1 (ES1), the first **official standard for JavaScript** (ECMAScript is the standard, JavaScript the language in practice);

2009

- 👉 ES5 (ECMAScript 5) is released with lots of great new features;

2015

- 👉 ES6/ES2015 (ECMAScript 2015) was released: **the biggest update to the language ever!**
- 👉 ECMAScript changes to an **annual release cycle** in order to ship less features per update 🙏

2016 – ∞

- 👉 Release of ES2016 / ES2017 / ES2018 / ES2019 / ES2020 / ES2021 / … / ES2089 😅

BACKWARDS COMPATIBILITY: DON'T BREAK THE WEB!

```
// ES1 Code
function add(n) {
  var x = 5 + add.arguments[0];
  return x;
}
```

1997

👍

BACKWARDS COMPATIBLE

Modern JavaScript Engine

2020

DON'T BREAK THE WEB!

- 👉 Old features are **never** removed;
- 👉 Not really new versions, just **incremental updates** (releases)
- 👉 Websites keep working **forever**!

Modern JavaScript Engine

2020

👎

NOT FORWARDS COMPATIBLE

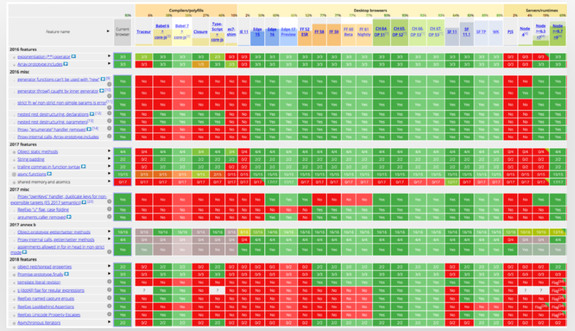
```
// ES2089 Code 😂
c int add n <=> int 5 + n
```

2089

HOW TO USE MODERN JAVASCRIPT TODAY

👩‍💻 **During development:** Simply use the latest Google Chrome!

🚀 **During production:** Use Babel to transpile and polyfill your code (converting back to ES5 to ensure browser compatibility for all users).



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

ES5

- 👉 Fully supported in all browsers (down to IE 9 from 2011);
- 👉 Ready to be used today 👍

BABEL

ES6/ES2015 ↓ ES2020

- 👉 **ES6+**: Well supported in all **modern** browsers;
- 👉 No support in **older** browsers;
- 👉 Can use **most** features in production with transpiling and polyfilling 😃

ES2021 – ∞

- 👉 **ESNext**: Future versions of the language (new feature proposals that reach Stage 4);
- 👉 Can already use **some** features in production with transpiling and polyfilling.

Will add new videos

(As of 2020)

MODERN JAVASCRIPT FROM THE BEGINNING

🔥 Learn **modern JavaScript from the beginning!**

☝️ But, also learn how some things used to be done **before** modern JavaScript (e.g. `const` & `let` vs `var` and function constructors vs ES6 `class`).

3 reasons why we should not forget the Good Ol' JavaScript:

👉 You will better understand how JavaScript actually works;

👉 Many tutorials and code you find online today are still in ES5;

👉 When working on old codebases, these will be written in ES5.