front-end

JS Basics

lab 1/8

Introduction

- Robert Spier
- Oud-CMD
- Doet dingen met data visualisatie
- Runt een eigen muzieklabel ernaast
- Houdt van cappuchino met suiker

- Man van **de klok**
- Neem je **eigen verantwoordelijkheid**
- **Show**, don’t tell
- Als je er bent, **dan lever je**

today

I. Course (recap)

II. Progressive Enhancement

III. JavaScript basics (ES5/ES6)

IV. JavaScript datatypes

Course

course

description

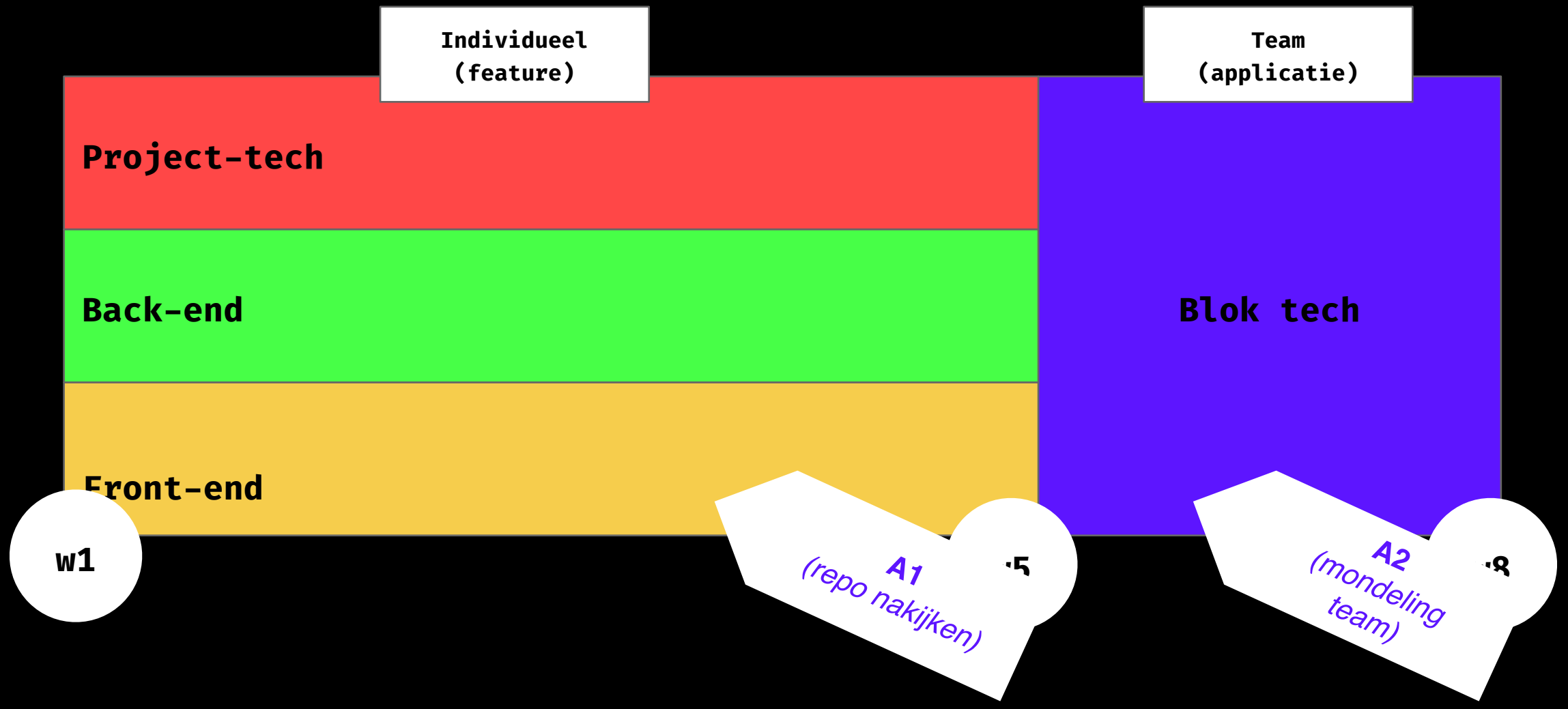
In Front-end 2 we concentrate **on improving the knowledge core JavaScript concepts** and learn how to **progressively enhance interfaces** with mostly JavaScript. Multiple frontend components are designed and build using client-side JavaScript.

/readme.md

course

goals

- You improve your knowledge about core JavaScript concepts
- You are able to build progressively enhanced frontend components
- You can build a web application with semantic HTML/CSS/JS
- You can write docs and explain your code and application structure
- You are able to research sources and read documentation


Individueel
(feature)
Team
(applicatie)
Project-tech
Back-end
Front-end
Blok tech
w1
A1
(repo nakijken)
A2
(mondeling
team)

Lectures

Work

- Lectures provide you with valuable information regarding FeD subjects
- Read some relevant theory prior to the lab
- Work on codepen exercises to apply the theory
- Work on your client-side progressive enhancement feature

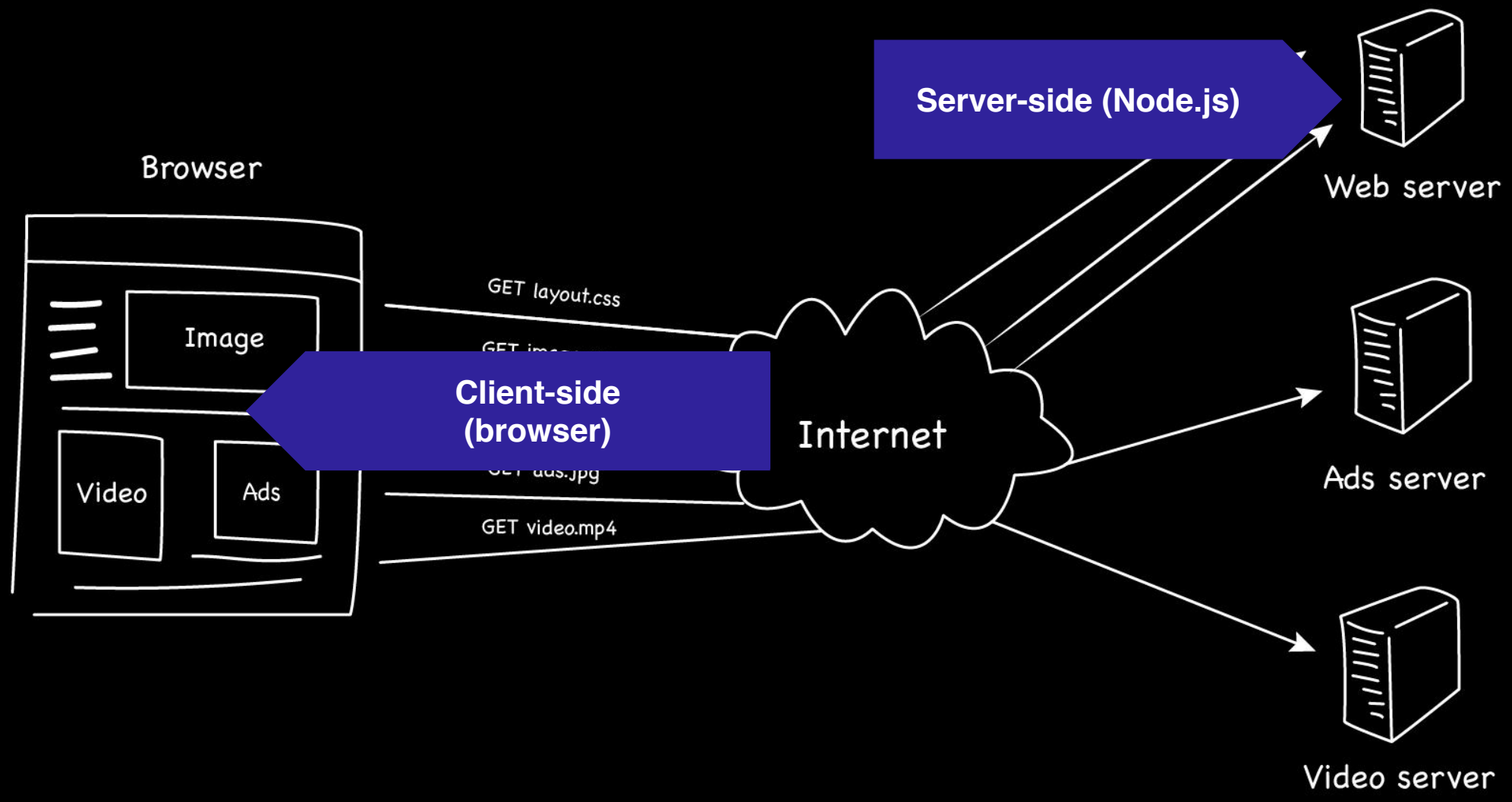
Progressive enhancement

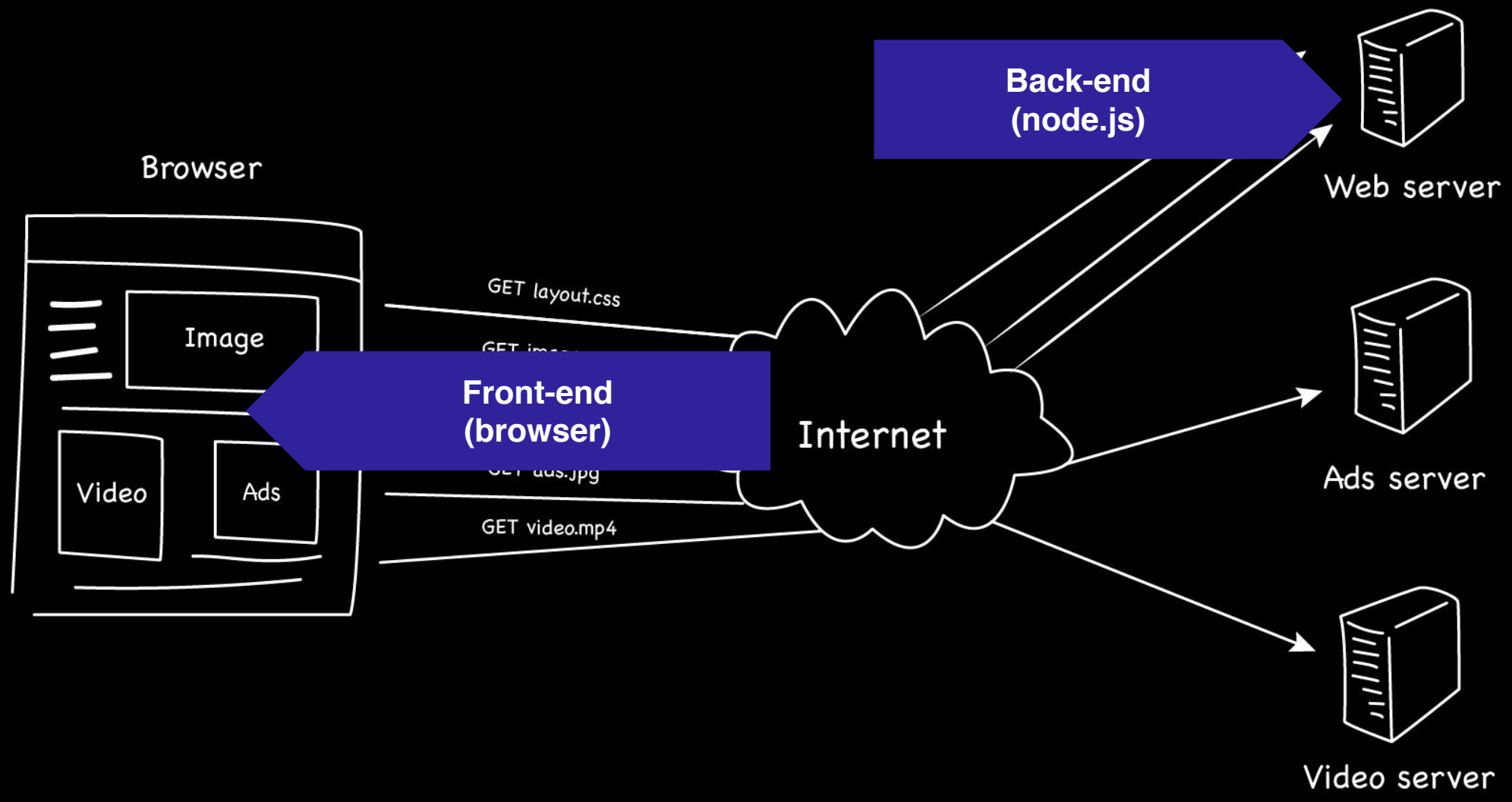
Assessment

description

For your A1 assessment, you're going to implement a *progressively enhanced component.* In short, you're going to enhance the client-side experience of the user by doing research, documenting patterns and implement the principle of *progressive enhancement* using JavaScript.

/readme.md


Server-side (Node.js)
Browser
Image
Video
Ads
Client-side
(browser)
GET layout.css
GET video.mp4
Internet
Web server
Ads server
Video server


Back-end
(node.js)
Browser
Image
Video
Ads
Front-end
(browser)
GET layout.css
GET video.mp4
Internet
Web server
Ads server
Video server

Enhancement

Definition

The word progressive in progressive enhancement means creating a design that achieves a simpler-but-still-usable experience for users of older browsers and devices with limited capabilities, while at the same time being a design that **progresses the user experience up to a more-compelling, fully-featured experience for users of newer browsers and devices with richer capabilities.**

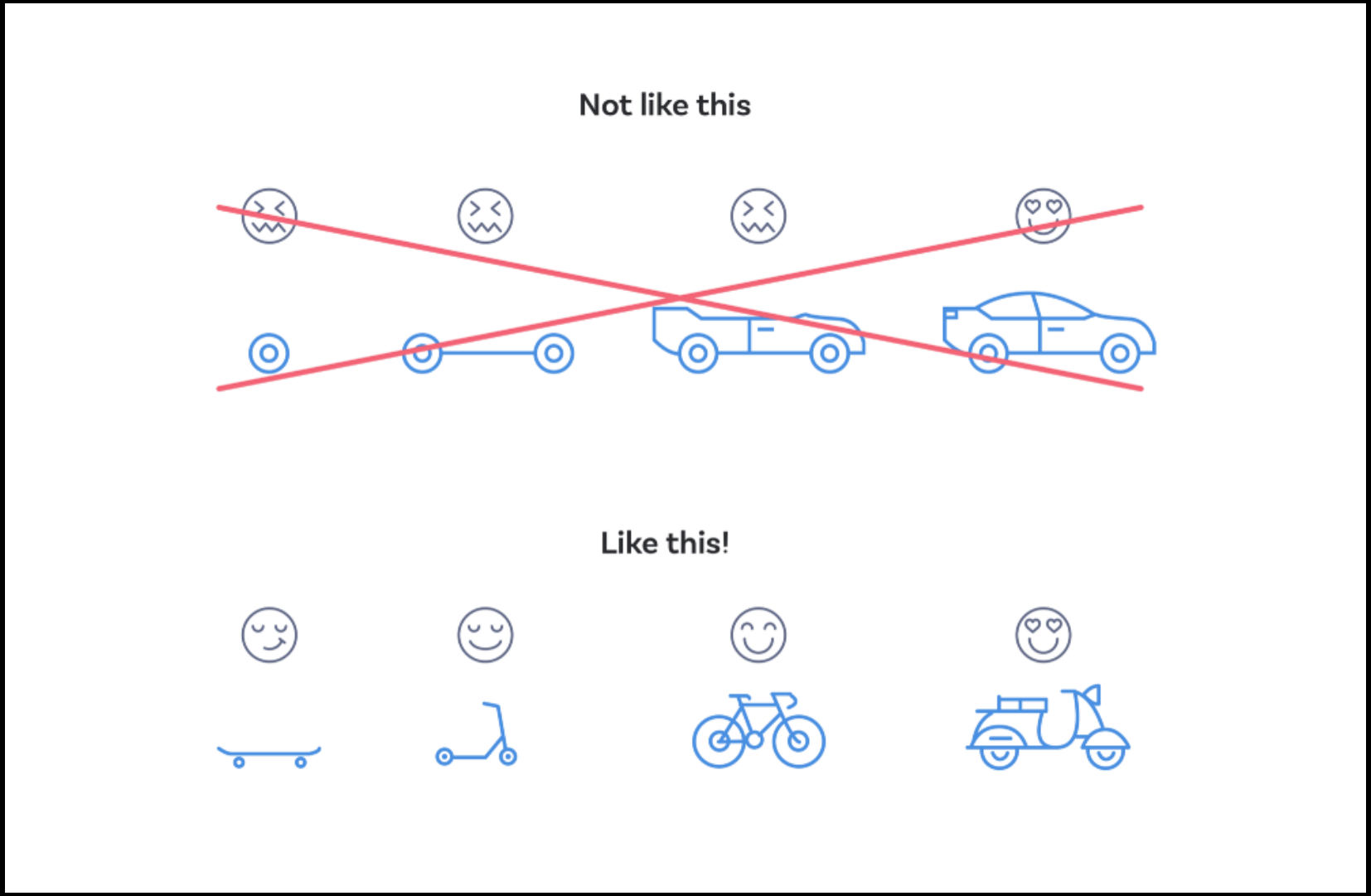
Mozilla MDN

Enhancement

Definition

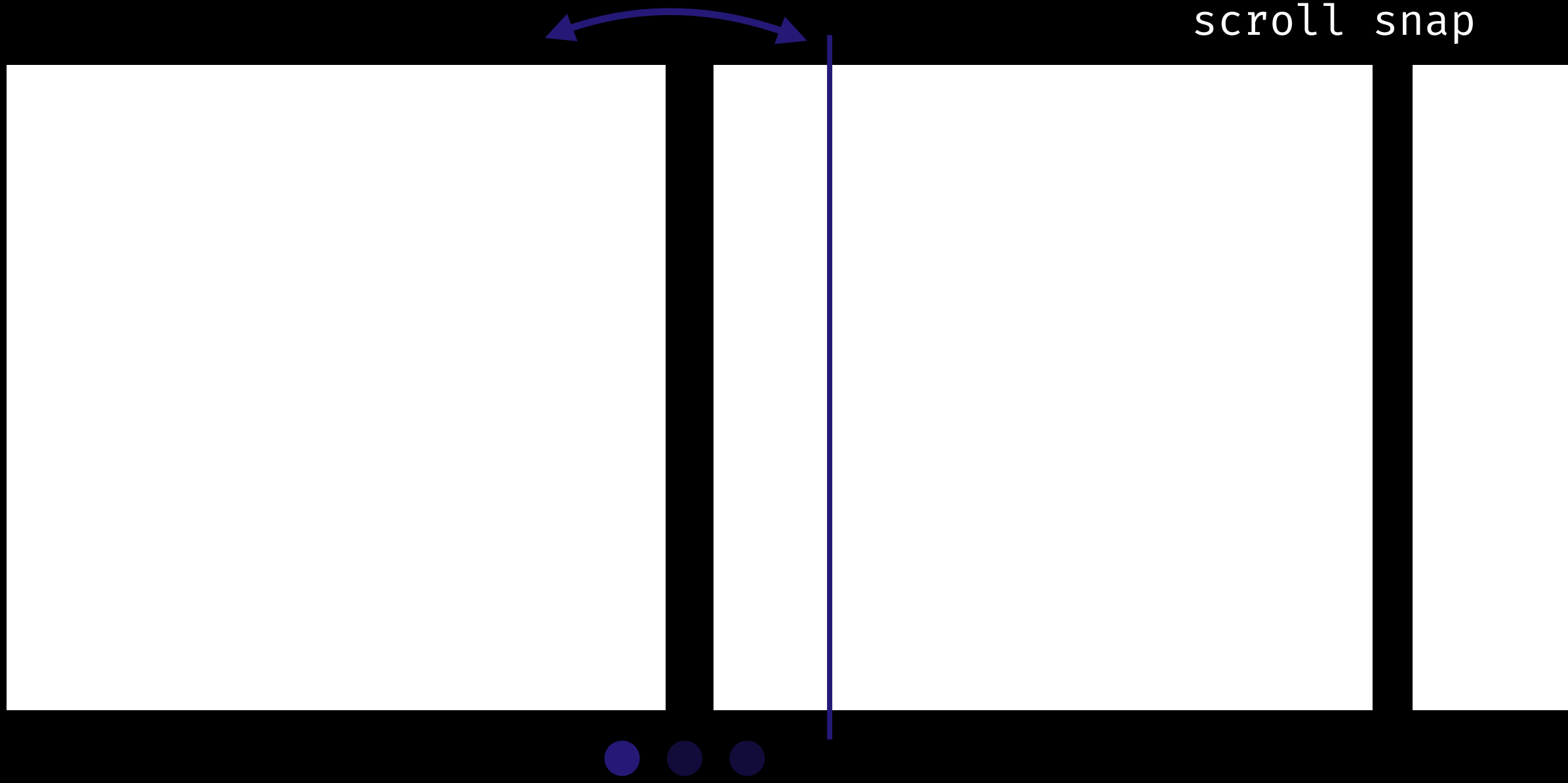
With progressive enhancement, every user has **their own experience of the site**, rather than an experience that the designers and developers demand of them.

Mozilla MDN



Andy Bell – The power of enhancement

scroll overflow


scroll snap

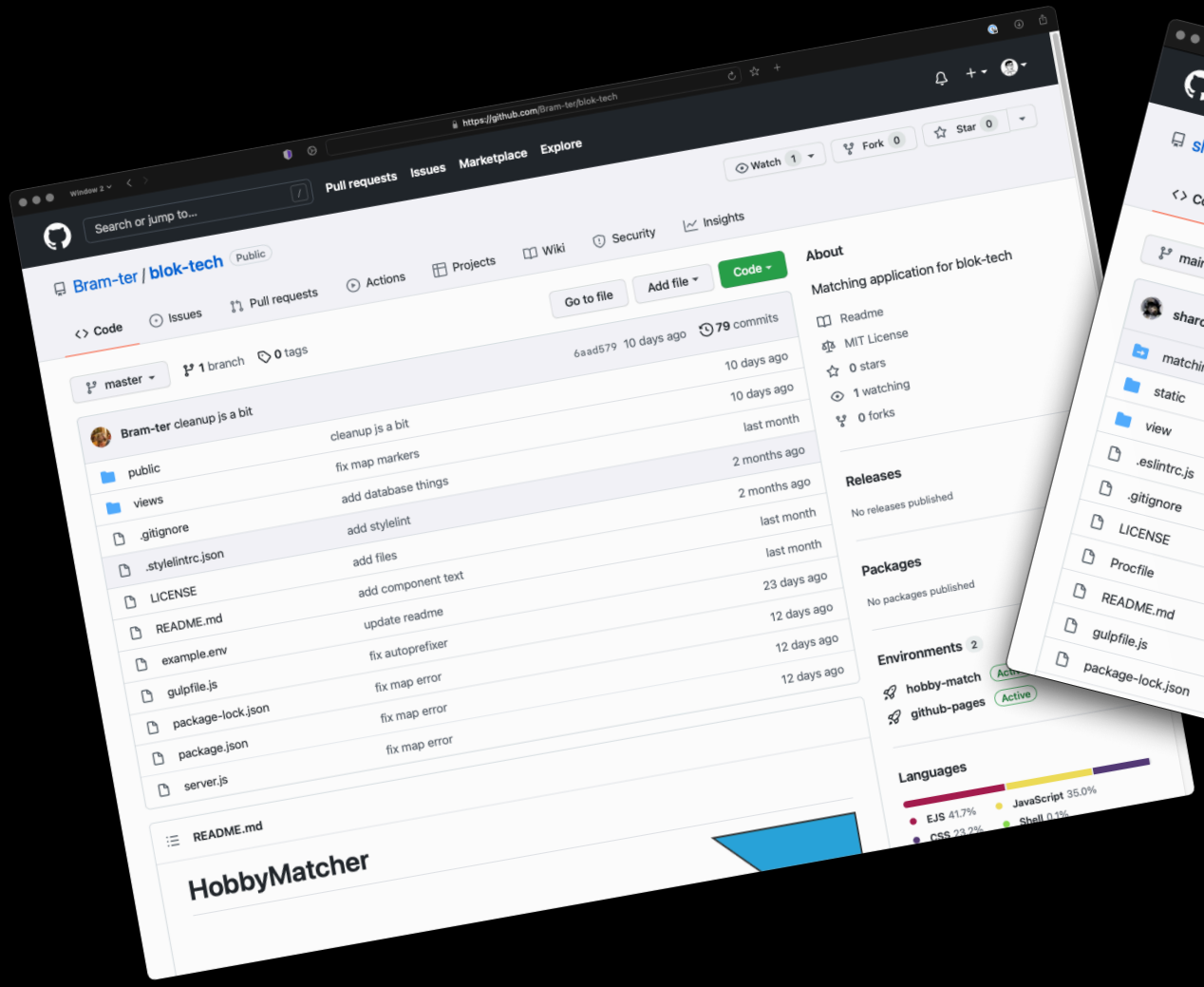
Eventlisteners

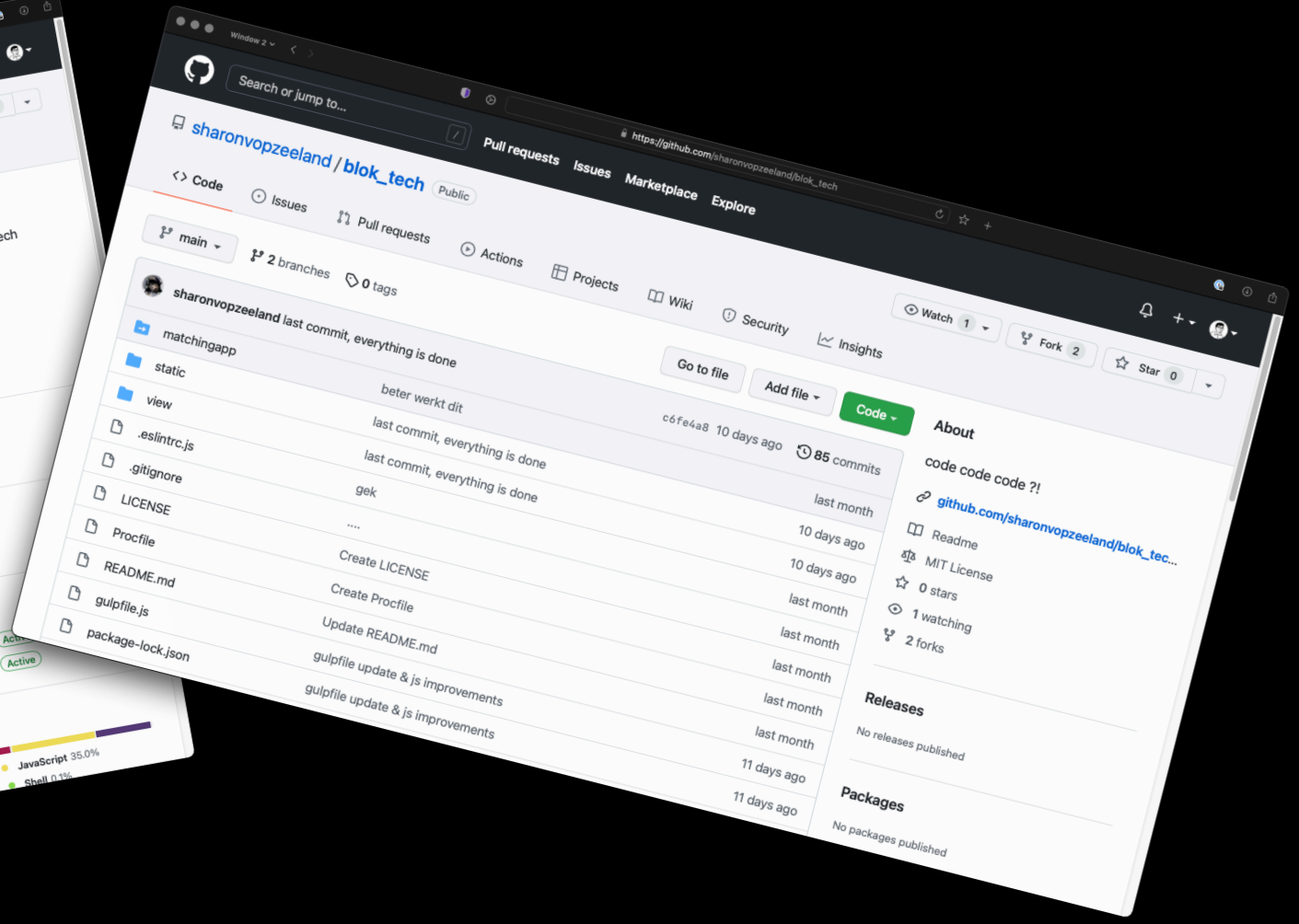
Intersection observer

Timeout?

Enhancement

student examples





assignment (+25m)

A wild progressive enhancement appeared! Search for a progressively enhanced component inside a web application (or website for that matter) that you use on a daily basis.

- *What type of component is it (design pattern)*
- *How is it enhanced with CSS?*
- *What client-side JavaScript is used?*
- *Does it use a web api?*
- *Does it work with JavaScript disabled?*

Break!

JavaScript Basics

JavaScript?

jit?

JavaScript (JS) is a **lightweight interpreted or JIT-compiled** programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js.

developer.mozilla.org

JS

Ecmascript?

?

Search...

TABLE OF CONTENTS

ECMA-262, 12th edition, June 2021
ECMAScript® 2021 Language Specification

ecma INTERNATIONAL

About this Specification

The document at https://tc39.es/ecma262/ is the most accurate and up-to-date ECMAScript specification. It contains the content of the most recent yearly snapshot plus any finished proposals (those that have reached Stage 4 in the proposal process and thus are implemented in several implementations and will be in the next practical revision) since that snapshot was taken.

Contributing to this Specification

This specification is developed on GitHub with the help of the ECMAScript community. There are a number of ways to contribute to the development of this specification:

GitHub Repository: https://github.com/tc39/ecma262
Issues: All Issues, File a New Issue
Pull Requests: All Pull Requests, Create a New Pull Request
Test Suite: Test262
Editors:
- Jordan Harband (@ljharb)
- Shu-yu Guo (@_shu)
- Michael Ficarra (@smooshMap)
- Kevin Gibbons (@bakkoting)

Community:
- Discourse: https://es.discourse.group
- IRC: #tc39 on freenode
- Mailing List Archives: https://esdiscuss.org/

Refer to the colophon for more information on how this document is created.

Introduction

This Ecma Standard defines the ECMAScript 2021 Language. It is the twelfth edition of the ECMAScript Language Specification. Since publication of the first edition in 1997, ECMAScript has grown to be one of the world's most widely used general-purpose programming languages. It is best known as the language embedded in web browsers but has also been widely adopted for server and embedded applications.

ECMAScript is based on several originating technologies, the most well-known being JavaScript (Netscape) and JScript (Microsoft). The language was invented by Brendan Eich at Netscape and first appeared in that company's Navigator 2.0 browser. It has appeared in all subsequent browsers from Netscape and in all browsers from Microsoft starting with Internet Explorer 3.0.

The development of the ECMAScript Language Specification started in November 1996. The first edition of this Ecma Standard was adopted by the Ecma General Assembly of June 1997.

ECMA International

JS

ES5

< 2015

JS

ES6

2015 >

ES2021

JS

ES2022

ES5

```
/* Select button and menu in DOM */
var button = document.querySelector("header nav button");
var menu = document.querySelector("header nav");

/* Add Event Listener to button */
button.addEventListener("click", openMenu);

/* Function open menu */
function openMenu() {
  menu.classList.toggle("open-menu");
}
```

ES6

```
/* Select button and menu in DOM */
const button = document.querySelector(`header nav button`);
const menu = document.querySelector(`header nav`);

/* Add Event Listener to button */
button.addEventListener(`click`, openMenu);

/* Function open menu */
const openMenu = () => {
  menu.classList.toggle(`open-menu`);
}
```

ES6

```
/* Select button and menu in DOM */
const button = document.querySelector(`header nav button`);
const menu = document.querySelector(`header nav`);

/* Add Event Listener to button */
button.addEventListener(`click`, openMenu);

/* Function open menu */
const openMenu = () => {
  menu.classList.toggle(`open-menu`);
}
```

ES6

```
var banaan = "banaan";

let appel = "appel";

const kiwi = "kiwi";
```

ES6

```
var banaan = "banaan";

let appel = "appel";

const kiwi = "kiwi";

/* Dit kan niet! */
kiwi = "manderijn";

/* Dit kan wel! */
appel = "perzik";
```

```
const button = document.querySelector('header nav button');

button.addEventListener('click', (element) => {
  console.log(element) // I log the HTML button element to the console!
})
```

```javascript
const docenten = ["Robert", "Danny", "Sonja", "Ivo", "Janno"]

/* Map is een array function waarmee we kunnen "loopen"
   over de waardes in de array hierboven */
docenten.map(element => {
  /* We loggen nu ieder los element naar de console */
  console.log(element);
})

/* Maar het kan zelfs nog sneller! */
docenten.map(element => console.log)
```

```
const button = document.querySelector('header nav button');

button.addEventListener('click', (element) => {
  console.log(element) // I log the HTML button element to the console!
})
```

```
app.get('/', (req, res) => {
  res.send('root')
})

app.get('/', function (req, res) {
  res.send('root')
})
```

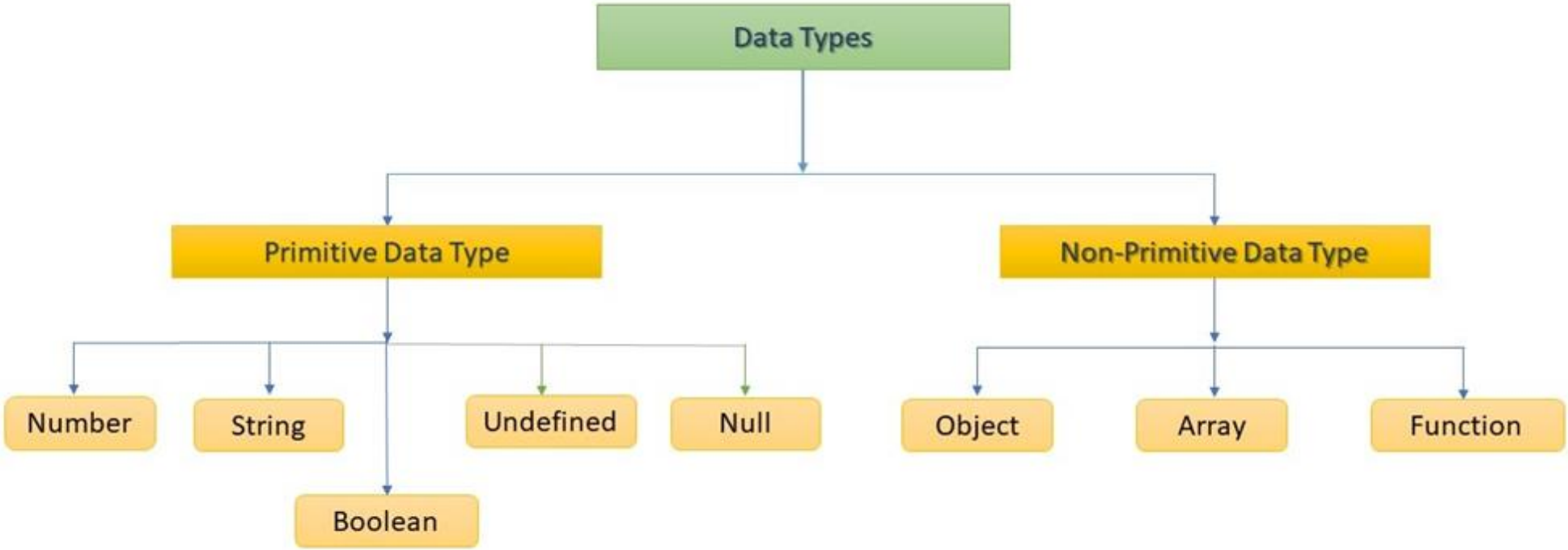
assignment

Rewrite the code you wrote for front-end (blokweb) to ES6 syntax.

OR

Work on the codepen about ES5 versus ES6. The goal is to re-write ES5 code to ES6.

JavaScript Data types


Data Types
Primitive Data Type
Non-Primitive Data Type
Number
String
Boolean
Undefined
Null
Object
Array
Function

```
const aString = "A string is variable between quotation marks"

const aBoolean = false || true // True or False. (nothing in between)

const aNumber = 0 // Any number, preferably rounded.
```

```
const aString = "A string is variable between quotation marks"

const aBoolean = false || true // True or False. (nothing in between)

const aNumber = 0 // Any number, preferably rounded.
```

```
// I exist but I am nothing. I'll yield a falsy result!
const aNull = null;

// I can hard-code an undefined variable, but variables that are
// not defined will also yield "undefined"
const aUndefined = undefined
```

```
const teachers = [
  'Robert',
  'Danny',
  'Sonja',
  'Ivo',
  'Janno'
]
```

```
const random = ["test", "bier", 382, false, true, { hello: "world" }, ['1', '2']];
```

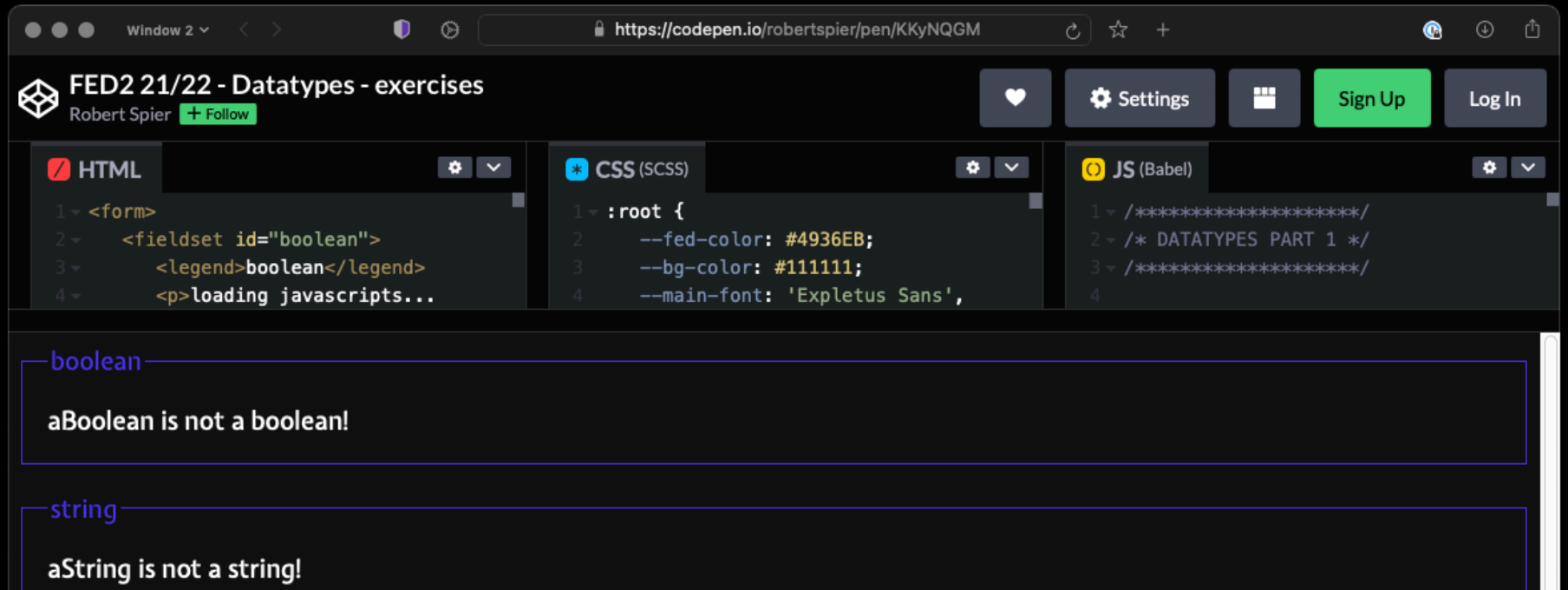
```javascript
const teacher = {
  name: "Robert",
  age: 29,
  location: ["52.359135", "4.909953"],
  hasCar: false
}

console.log(teacher.name) // Yields "Robert"
console.log(teacher['age']) // Yields "29"
```

```javascript
const teacher = {
  name: "Robert",
  age: 29,
  location: ["52.359135", "4.909953"],
  hasCar: false,
  sayHi: function() { // We use function() instead of => to preserve context!
    console.log(this) // "this" now points to this object as a keyword!
    console.log(`Hi ${this.name}!`)
  }
}

console.log(teacher.sayHi()) // Yields "Hi Robert!"
```

```javascript
function secret(thing) {
  console.log(`It's a secret ${thing}`);
}

secret('hat!') // Yields: It's a secret hat!
```

assignment

Work on the codepen about Datatypes. The goal is to declare several different data types and access the data.



- FreeCodeCamp Basics JavaScript
- JavaScript 30 by Wes Bos

Enhancement

enhancement

> Research progressive enhancement and document in your wiki your idea of the progressively enhanced component you want to build during this project.

Synopsis

- **Time**: 6:00h
- **Goals**: subgoal 1, subgoal 2
- **Due**: before week 2

Assignment

Research progressive enhancement and document your idea of the component you want to build during this project.

- Look for inspiration, what components do you see in the wild and will fit your matching application?
- What will be done server-side, what will be done client-side?
- Are you going to use web API's on the client to enhance the user experience?
- What will happen if JavaScript is disabled? Will it degrade gracefully to HTML & CSS?

Read for week 2:

- MDN. (2021, 14 september). Introduction to the DOM - Web APIs
- MDN. (2022, 4 februari). Introduction to events - Learn web development

Additional resources

work on week-1

exit;

see you in **lab-2**!