Toegepaste Informatica

1TX

2022-2023

UCLL HOGESCHOOL

Front-end development

Part 2: Dynamic web applications

1. Javascript Introduction

J. Van Hee, J. Pieck, G. Jongen, A. Vranken

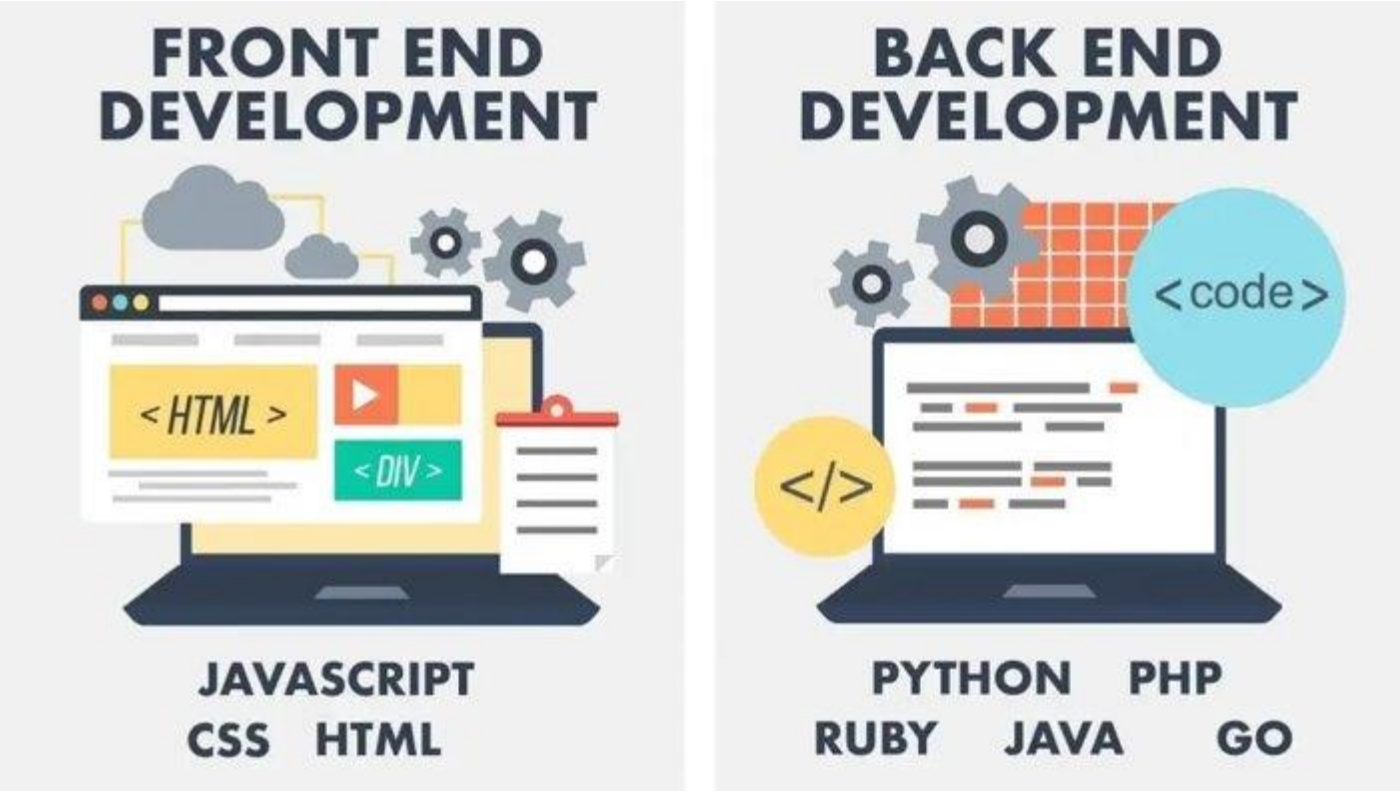
Javascript: Introduction

- Static vs dynamic websites
- Front-end vs Back-end development
- What is Javascript?
- Using Javascript

Static vs dynamic websites

- Static websites:
  - All users see the same, stable content.
  - Faster to setup, more content management work.
  - Page-by-page navigation.
- Dynamic websites:
  - Content changes with user interaction.
  - More initial (programming) work, more efficiënt to manage content.
  - Parts of a webpage can change without browser refresh.

Front-end vs back-end development



What is Javascript?

- Cross-platform programming language.
- Supported by every browser.
- Client-side javascript:
  - In browser
  - Makes websites interactive: animations, clicking on HTML elements,…
  - Manipulate the **Document Object Model** (DOM).
  - Fetch data from a server to display to users.
- Server-side javascript:
  - Stand-alone, without a browser (with a library like **NodeJS)**.
  - Develop a back-end application (database queries, file manipulation,…).

Using Javascript

- Inline in HTML code with **<script>** tag
- Externalized in a ***.js** file:
  - Preferred method: clean code, more maintainable
- Import scripts in **<head>**:

```
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>The Farm</title>
  <link type="text/css" href="style/style.css" rel="stylesheet" />
  <script type="text/javascript" src="js/dom.js" defer></script>
  <script type="text/javascript" src="js/farm.js" defer></script>
</head>
```

- Defer: the script is executed after the whole page is loaded
  - Important when DOM manipulation happens in the script
- Order of importing scripts is important
  - Eg. Calling functions in **farm.js** which are defined in **dom.js**

First: Using browser developer tools

- Learn to use the developer console of your browser to
  - Debug code, detect errors
  - Experiment: write, execute and test Javascript code

Javascript: Basic syntax

- Variables
- Datatypes
- Operators
- Conditionals
- Functions
- Objects

Variables

- Declaring

```
let myName;
let myAge;
```

- Assigning

```
let myName = ‘John’;
myName = ‘Frank’;
let myAge = 45;
myAge = 25;
```

Variables

- Constants
  - Can’t be assigned a new value after initialization

```
const myName = ‘John’;
myName = “Frank”;     // error
```

  - Prefer using *const* above *let,* as it results in cleaner code. Since you can actually change the datatype in JS with a new assignment, this can result in unpredictable results.

Datatypes

- Variable types

```
let myName = ‘John’;                          // string
let myAge = 45;                               // number (integers and floats)
let isOld = false;                            // boolean
let children = [‘Tim’, ‘Suzy’]                // array
let identity = {name: ‘John’, age: 45}        // object
```

- Dynamic typing: in JS, the type of a variable **can change**

```
let info = ‘John’;    // string
info = 45;            // number
```

Operators

- Arithmetic

```
3 + 8.1;        // 11.1
1 / 3;          // 0.3333333333333333
14 % 3;         // modulo = 2
2 ** 8          // power = 256
```

- String concatenation

```
let myName = “John”;
let myAge = 45;
let greeting = ‘Name: ’ + myName + ‘ Age: ’ + myAge;
let greeting = ´Name: ${myName} Age: ${myAge}´ // Concatenation with template literals
```

Conditionals

- Comparing

```
2 < 3          // True
2 <= 3         // True
2 > 3          // False
3 >= ‘2’       // True: string is first converted to number
```

- Strict equality

```
3 == ‘3’       // True
3 === ‘3’      // False
```

- Always use strict equality!

Conditionals

- If / then / else

```
if (myAge < 18) {
  console.log('Child');
} else if (myAge >= 18 && myAge < 67) {
  console.log('Adult');
} else {
  console.log('Retired');
}
```

- Logical operators

```
true && false // false
true || false // true
```

Conditionals

- Conditional ternary operator

```
console.log(myAge < 18 ? 'Child' : 'Adult');
```

Functions

- Classic

```
function calculateSum(a, b) {
  return a + b;
}
…
let s = calculateSum(2,4);
```

- Modern: arrow functions

```
const calculateSum = (a, b) => {
  return a + b;
}
…
let s = calculateSum(2,4);
```

Functions

- Shorthand version, if function contains 1 line:

```
const calculateSum = (a, b) => a + b;
…
let s = calculateSum(2,4);
```

- Try to use JS as a “functional” language and express your functionality with functions (even if they are only 1 line).
- In JS, there’s a lot you can do with functions (assign them to variables, pass them as parameters). More in next lessons.

Objects

- Declaring and using objects

```
const identity = { name: ‘John’, age: 45 }
console.log(`Hello ${identity.name} who is ${identity.age} years old.`)
```

- Destructuring object fields in variables

```
const identity = { name: ‘John’, age: 45 }
const { name, age } = identity
console.log(`Hello ${name} who is ${age} years old.`)
```