

# Programming for the web

An introduction to front-end development with React

Rens Holmer

July 7, 2024

[illegible][illegible][illegible]

WAGENINGEN  
UNIVERSITY & RESEARCH

# Goal of this session

- ▶ Introduction to building *interactive* websites

## Activities

- ▶ Familiarize yourself with main building blocks of a website
- ▶ Learn about modern toolchains and frameworks for the web
- ▶ Build your own interactive bioinformatics data visualization

## Material

```
git clone https://github.com/holmrenser/introduction_to_react
```

# Assignment 1

Explore two versions of a simple static website:

`example/simple_site`

`example/simple_site_single_file`

- ▶ Open the workshop folder in an IDE (e.g. VSCode)
- ▶ Open `index.html` in a web browser
- ▶ Open the developer tools (`cmd + shift + C`)
- ▶ Click the button
- ▶ Explore the code
- ▶ *Discuss*

# Some technical background

## Core components

- ▶ **HTML:** HyperText Markup Language, i.e. text with links and semantics for 'function' such as headers or links.
- ▶ **JS:** JavaScript, actions, interaction, Document Object Model (DOM).
- ▶ **CSS:** Cascading Style Sheets, visual styling. Style applies to HTML tag and all it's children (hence 'cascading').

## Key concepts

- ▶ Asynchronous programming: networks can be slow, local computation should not wait (examples in next slides)
- ▶ Using the browser as a debugger and REPL

# We have to talk about AJAX

- ▶ Aynchronous Javascript And Xml
- ▶ Originally introduced as concept for typical web tasks
- ▶ Does not actually have to use XML (today we use JSON)
- ▶ Modern JS: implemented in e.g. `fetch` API



# Javascript quirks (variables)

Many aspects of JS are superficially similar to Python

```
./variables.py
```

```
1 a = "hi"
2 b = 2
3 c = 2.0
4
5 b + c
6 # 4.0
7
8 b = a # Overwrite b
9 d = "hello"
10 b + d
11 # "hi hello"
12
13 c + d
14 # TypeError: unsupported
15 # operand type(s) for +:
16 # 'int' and 'str'
```

Python

```
./variables.js
```

```
1 const a = "hi";
2 let b = 2;
3 const c = 2.0;
4
5 b + c;
6 // 4
7
8 b = a; // Overwrite b
9 const d = "hello";
10 b + d;
11 // "hi hello"
12
13 c + d;
14 // "2 hello"
```

Javascript

# Javascript quirks (functions)

```
./function.py
1 def select_second_letter(string):
2     return string[1]
3
4
5 select_third_letter = lambda string: string[2]
6
7 greeting = "Hello!"
8 select_second_letter(greeting)
9 # "e"
10
11 select_third_letter(greeting)
12 # "l"
```

Python

```
./function.js
1 function selectSecondLetter(string) {
2     return string.slice(1, 2);
3 }
4
5 const selectThirdLetter = (string) => string.slice(2, 3);
6
7 const greeting = "Hello!";
8 selectSecondLetter(greeting);
9 // "e"
10
11 selectThirdLetter(greeting);
12 // "l"
```

Javascript



# Javascript quirks (classes)

```
./class.py
1 class Sequence:
2     def __init__(self, header, seq):
3         self.header = header
4         self.seq = seq
5
6     def reverse(self):
7         return Sequence(
8             self.header,
9             self.seq[::-1]
10        )
11
12
13 seq = Sequence("test", "ACGTA")
14 seq.reverse()
15 # <__main__.Sequence object at 0x1011f5750>
```

Python

```
./class.js
1 class Sequence {
2     constructor(header, seq) {
3         this.header = header;
4         this.seq = seq;
5     }
6
7     reverse() {
8         return new Sequence(
9             this.header,
10            this.seq.split("").reverse().join("")
11        );
12    }
13 }
14
15 const seq = new Sequence("test", "ACGTA");
16 seq.reverse();
17 // Object { header: "test", seq: "ATGCA" }
```

Javascript

# Javascript quirks (operations)

● ● ● ./operations.py

```
1 1 == "1.0"
2 # False
3
4 1 != "1.0"
5 # True
6
7 1 + 1
8 # 2
9
10 1 + 1.0
11 # 2.0
12
13 12 / 5
14 # 2.4
15
16 12 // 5
17 # 2
```

Python

● ● ● ./operations.js

```
1 // equality operator
2 // attempts to convert
3 1 == "1.0";
4 // true
5
6 1 != "1.0";
7 // false
8
9 // strict equality operator
10 // does not convert
11 1 === "1.0";
12 // false
13
14 1 !== "1.0";
15 // true
16
17 1 + 1.0;
18 // 2
19
20 1 + "2.1";
21 // "12.1"
22
23 1 / "2.1";
24 // 0.47619047619047616
```

Javascript



WAGENINGEN  
UNIVERSITY & RESEARCH

# Javascript quirks (types)

```
./types.py

1 # Boolean
2 True
3 False
4
5 # String
6 "Hi"
7
8 # Integer
9 2
10
11 # Float
12 3.14
13
14 # List
15 [1, 2, "Hi", None]
16
17 # Tuple
18 (1, "Hi")
19
20 # Dictionary
21 {1: 2, "a": 2, (1, "Hi"): [2, 3, 3]}
```

Python

```
./types.js

1 // Boolean
2 const t = true;
3 const f = false;
4
5 // String
6 const string = "Hi";
7
8 // Number
9 const int = 3;
10 const float = 3.14;
11
12 // Array
13 const arr = [1, 2, "Hi"];
14
15 // Object
16 const obj = { 1: 2, b: 2, [[1, "Hi"]]: [1, 2, 3] };
17 /*
18 Some keys get converted!
19 --> Object { 1: 2, b: 2, "1,Hi": (3) [...] }
20 */
```

Javascript

# Javascript quirks (comments)

```
./comments.py  
1 # Single line comments  
2  
3 ""  
4 Multi line comments  
5 ""
```

Python

```
./comments.js  
1 // Single line comments  
2  
3 /*  
4 Multi line comments  
5 */
```

Javascript

# Javascript quirks (controlflow)

```
./controlflow.py
1 # https://en.wikipedia.org/wiki/Fizz_buzz
2 for i in range(20):
3     div_by_3 = i % 3 == 0
4     div_by_5 = i % 5 == 0
5     if div_by_3 and div_by_5:
6         print("Fizz Buzz")
7     elif div_by_3:
8         print("Fizz")
9     elif div_by_5:
10        print("Buzz")
11    else:
12        print(i)
```

Python

```
./controlflow.js
1 // https://en.wikipedia.org/wiki/Fizz_buzz
2 for (let i = 0; i < 20; i += 1) {
3     const divBy3 = i % 3 === 0;
4     const divBy5 = i % 5 === 0;
5     if (divBy3 && divBy5) {
6         console.log("Fizz Buzz");
7     } else if (divBy3) {
8         console.log("Fizz");
9     } else if (divBy5) {
10        console.log("Buzz");
11    } else {
12        console.log(i);
13    }
14 }
```

Javascript

# Javascript quirks (string formatting)

```
./string_formatting.py
```

```
1 a = 2 + 3  
2 f"Variable a contains {a}"  
3 # "Variable a contains 5"
```

Python

```
./string_formatting.js
```

```
1 const a = 2 + 3;  
2 `Variable a contains ${a}`;  
3 // "Variable a contains 5"  
4 // Note the backticks!
```

Javascript

# Modern JS

Increasingly large/complex designs result in modern framework approach: introduce template language that emphasizes modularity, 'transpile' to HTML, JS, CSS.

- ▶ Some common frameworks: React, Vue, Angular, Svelte, etc.
- ▶ transpiling
- ▶ component re-use
- ▶ state management
- ▶ AJAX

# Assignment 2

Build a React app for visualizing MSAs