Programming for the web

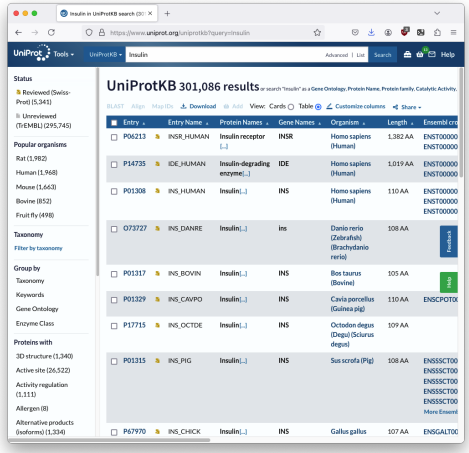
An introduction to front-end development with React

Rens Holmer

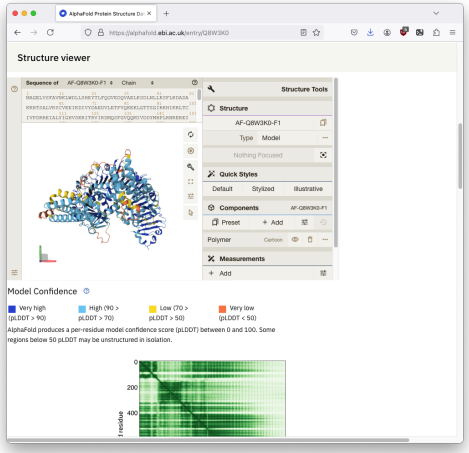
July 8, 2024

WAGENINGEN
UNIVERSITY & RESEARCH

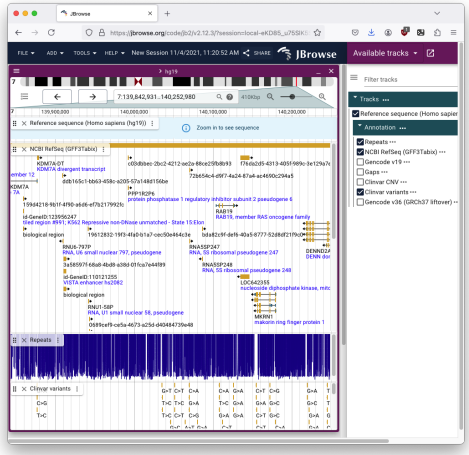
Web development in bioinformatics



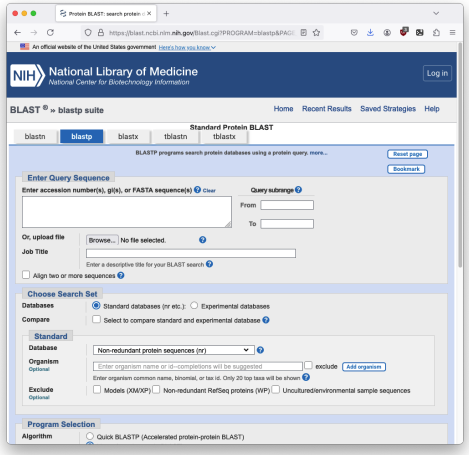
Uniprot



Alphafold-DB



JBrowse



NCBI BLAST

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 sematics 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

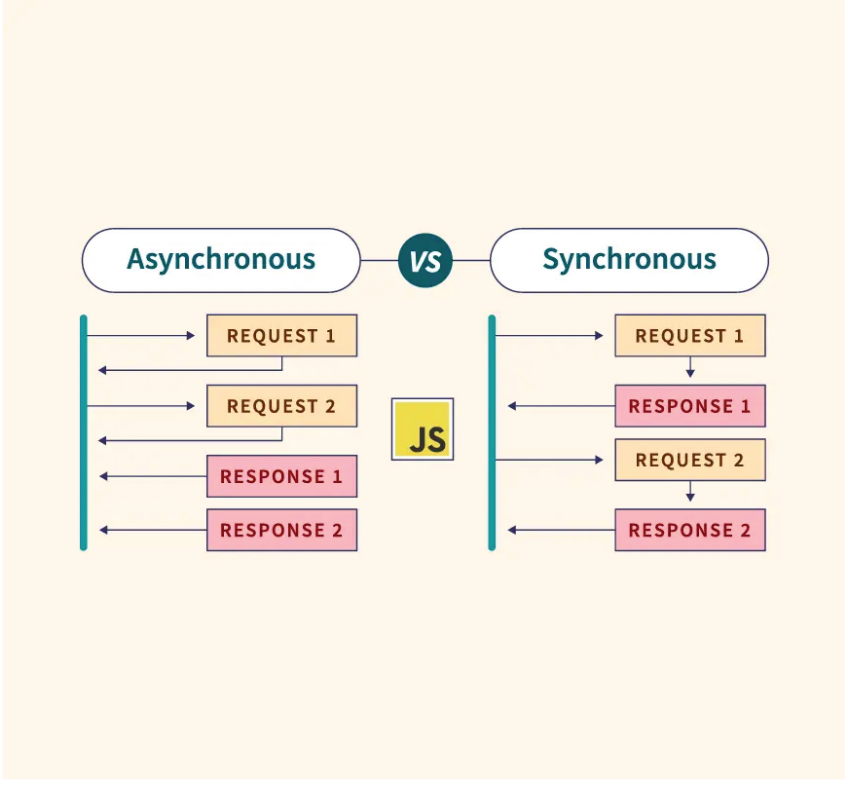
We have to talk about AJAX

- Asynchronous 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

The other AJAX

Event loop

On the web, we send requests and respond *asynchronously*



Javascript quirks (variables)

Many aspects of JS are superficially similar to Python

./variables.py

```python
a = "hi"
b = 2
c = 2.0

b + c
# 4.0

b = a  # Overwrite b
d = " hello"
b + d
# "hi hello"

c + d
# TypeError: unsupported
# operand type(s) for +:
# 'int' and 'str'
```

Python

./variables.js

```javascript
const a = "hi";
let b = 2;
const c = 2.0;

b + c;
// 4

b = a; // Overwrite b
const d = " hello";
b + d;
// "hi hello"

c + d;
// "2 hello"
```

Javascript

Javascript quirks (functions)

./function.py

```python
def select_second_letter(string):
    return string[1]


select_third_letter = lambda string: string[2]

greeting = "Hello!"
select_second_letter(greeting)
# "e"

select_third_letter(greeting)
# "l"
```

Python

./function.js

```javascript
function selectSecondLetter(string) {
  return string.slice(1, 2);
}

const selectThirdLetter = (string) => string.slice(2, 3);

const greeting = "Hello!";
selectSecondLetter(greeting);
// "e"

selectThirdLetter(greeting);
// "l"
```

Javascript

Javascript quirks (classes)

./class.py

```python
class Sequence:
    def __init__(self, header, seq):
        self.header = header
        self.seq = seq

    def reverse(self):
        return Sequence(
            self.header,
            self.seq[::-1]
        )


seq = Sequence("test", "ACGTA")
seq.reverse()
# <__main__.Sequence object at 0x1011f5750>
```

Python

./class.js

```javascript
class Sequence {
  constructor(header, seq) {
    this.header = header;
    this.seq = seq;
  }

  reverse() {
    return new Sequence(
      this.header,
      this.seq.split("").reverse().join("")
    );
  }
}

const seq = new Sequence("test", "ACGTA");
seq.reverse();
// Object { header: "test", seq: "ATGCA" }
```

Javascript

Javascript quirks (operations)

./operations.py

```python
1 == "1.0"
# False

1 != "1.0"
# True

1 + 1
# 2

1 + 1.0
# 2.0

12 / 5
# 2.4

12 // 5
# 2
```

Python

./operations.js

```javascript
// equality operator
// attempts to convert
1 == "1.0";
// true

1 != "1.0";
// false

// strict equality operator
// does not convert
1 === "1.0";
// false

1 !== "1.0";
// true

1 + 1.0;
// 2

1 + "2.1";
// "12.1"

1 / "2.1";
// 0.47619047619047616
```

Javascript

Javascript quirks (types)

./types.py

```python
# Boolean
True
False

# String
"Hi"

# Integer
2

# Float
3.14

# List
[1, 2, "Hi", None]

# Tuple
(1, "Hi")

# Dictionary
{1: 2, "a": 2, (1, "Hi"): [2, 3, 3]}
```

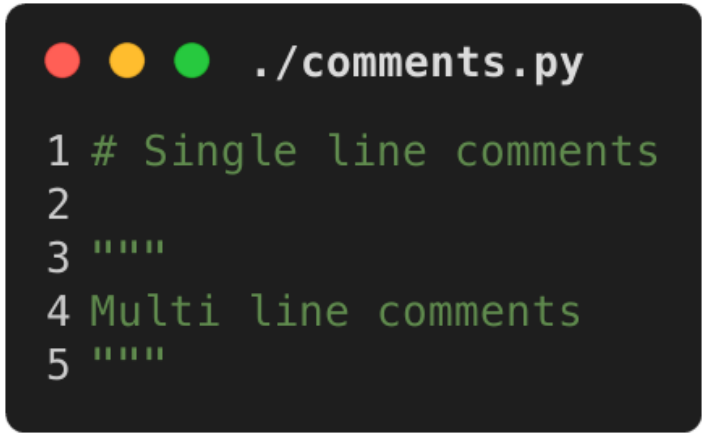
Python

./types.js

```javascript
// Boolean
const t = true;
const f = false;

// String
const string = "Hi";

// Number
const int = 3;
const float = 3.14;

// Array
const arr = [1, 2, "Hi"];

// Object
const obj = { 1: 2, b: 2, [[1, "Hi"]]: [1, 2, 3] };
/*
Some keys get converted!
--> Object { 1: 2, b: 2, "1,Hi": (3) […] }
*/
```

Javascript

Javascript quirks (comments)



Python

Javascript

Javascript quirks (controlflow)

./controlflow.py

```python
# https://en.wikipedia.org/wiki/Fizz_buzz
for i in range(20):
    div_by_3 = i % 3 == 0
    div_by_5 = i % 5 == 0
    if div_by_3 and div_by_5:
        print("Fizz Buzz")
    elif div_by_3:
        print("Fizz")
    elif div_by_5:
        print("Buzz")
    else:
        print(i)
```

Python

./controlflow.js

```javascript
// https://en.wikipedia.org/wiki/Fizz_buzz
for (let i = 0; i < 20; i += 1) {
  const divBy3 = i % 3 === 0;
  const divBy5 = i % 5 === 0;
  if (divBy3 && divBy5) {
    console.log("Fizz Buzz");
  } else if (divBy3) {
    console.log("Fizz");
  } else if (divBy5) {
    console.log("Buzz");
  } else {
    console.log(i);
  }
}
```

Javascript

Javascript quirks (string formatting)

./string_formatting.py

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

Python

./string_formatting.js

```javascript
const a = 2 + 3;
`Variable a contains ${a}`;
// "Variable a contains 5"
// 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/bundling
- Component re-use
- State management

Assigment 2

Build a React app for visualizing MSAs

- Explore created folder structure: `package.json`, `node_modules`, `vite.config.js`, `eslintrc.cjs`, `index.html`

```
conda create -n introduction-to-react nodejs
conda activate introduciton-to-react

npm create vite@latest my-app
# Select 'react' and 'javascript'

cd my-app

npm install
npm run dev
npm run build
```

Assignment 2 continued

Fetch some JSON data (available in the github repository of this tutorial)
Example URL:
`https://raw.githubusercontent.com/holmrenser/introduction_to_react/main/data/1.json`

```
fetch(url)
    .then(response ⇒ response.json())
    .then(msa ⇒ setMsa(msa));
```

Assignment 2 continued

Add an existing MSA component from `react-bio-viz`
`https://github.com/genenotebook/react-bio-viz`

```
// npm install --save-dev react-bio-viz

import { MultipleSequenceAlignment } from 'react-bio-viz'

<MultipleSequenceAlignment msa={msa} />
```

Assignment 2 continued

Responding to changing dependencies

```
import { useEffect } from 'react';

useEffect(() ⇒ {
  someCodeThatResponds( ... args);
}, [dependecies]);
```

Conclusions

- Modern JS focuses on re-usable components
- Toolchains help developers, obscure technical details