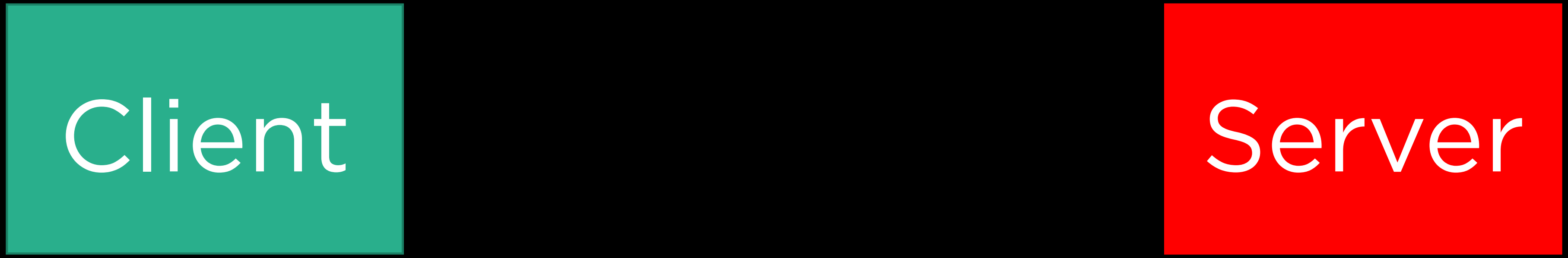
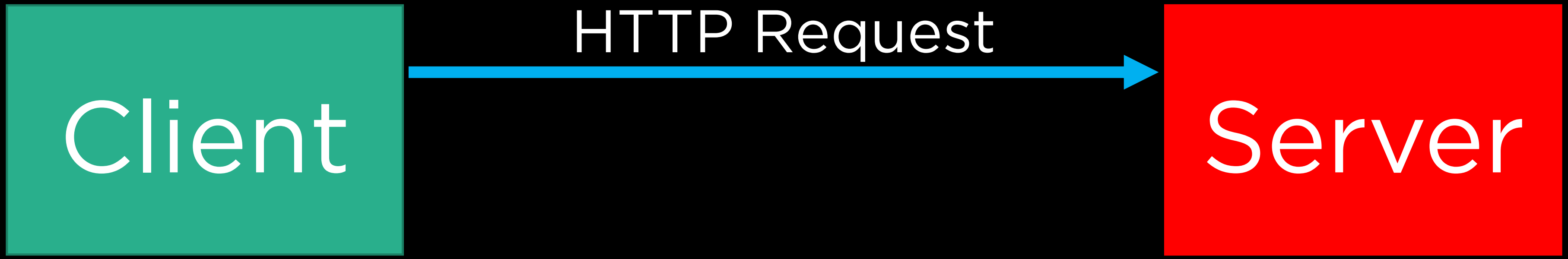
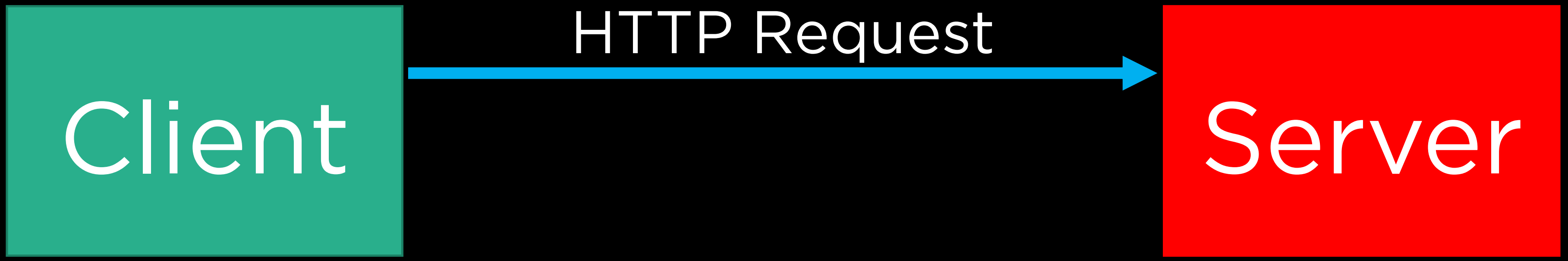
Software Engineering in the Arts and Humanities

JavaScript

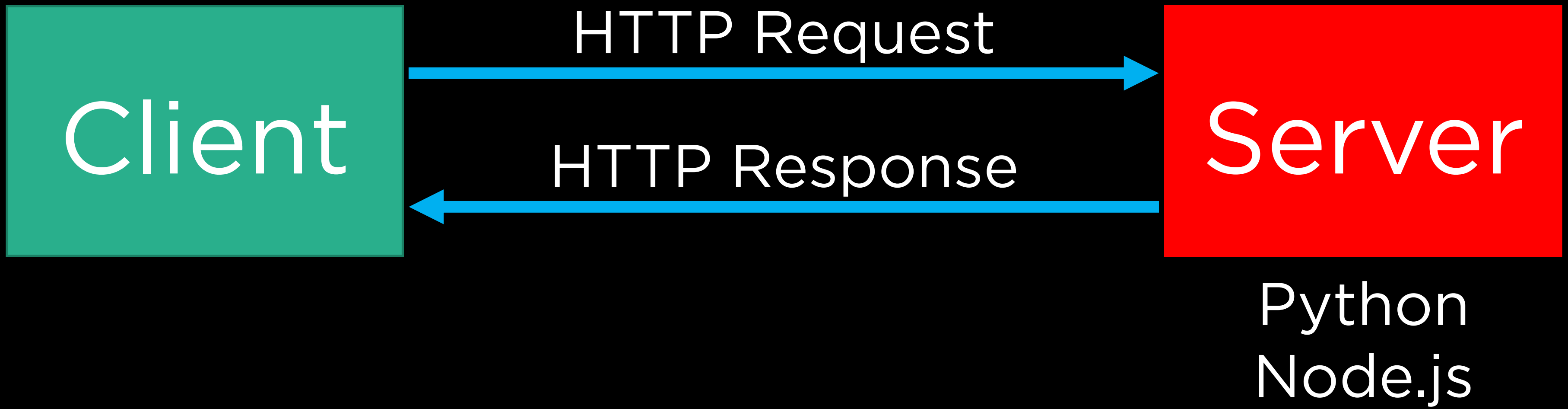
September 11, 2019

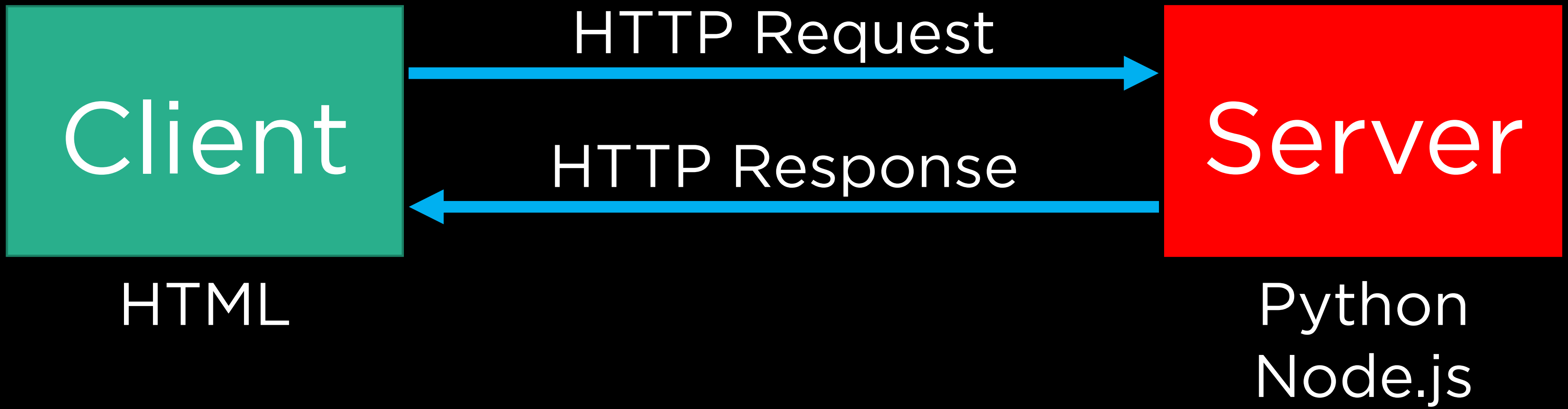

Client
Server

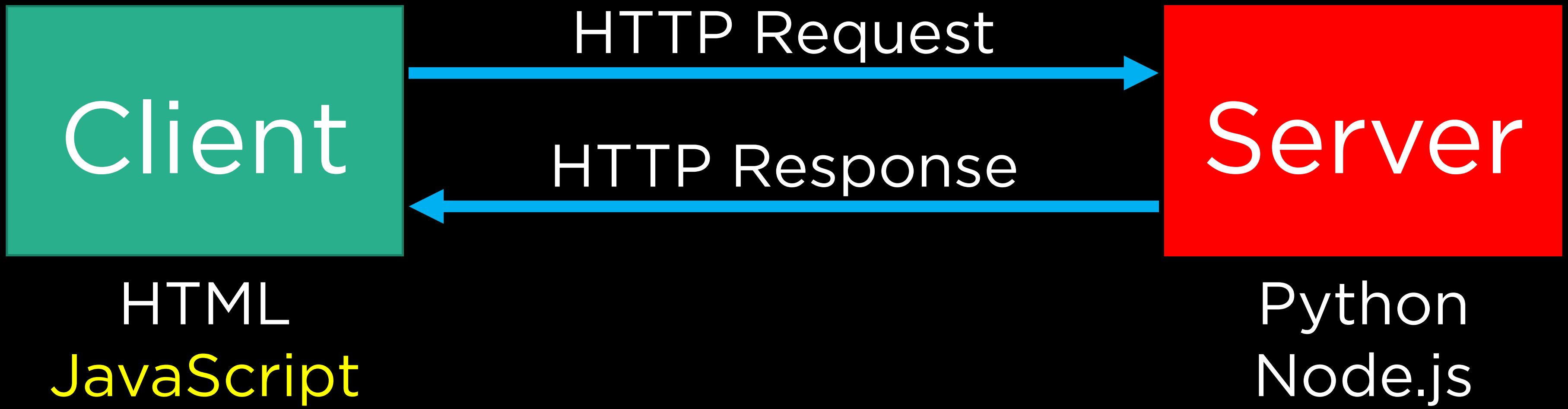

Client
HTTP Request
Server



Python
Node.js


Client
HTTP Request
HTTP Response
Server
Python
Node.js


Client
HTTP Request
HTTP Response
Server
HTML
Python
Node.js


HTTP Request
Client
Server
HTTP Response
HTML
JavaScript
Python
Node.js

JavaScript

JavaScript ES6

```
<script>
   alert('Hello, world!');
</script>
```

```
<script>
   alert('Hello, world!');
</script>
```

Functions

```
function hello() {
    alert('Hello, world!');
}
```

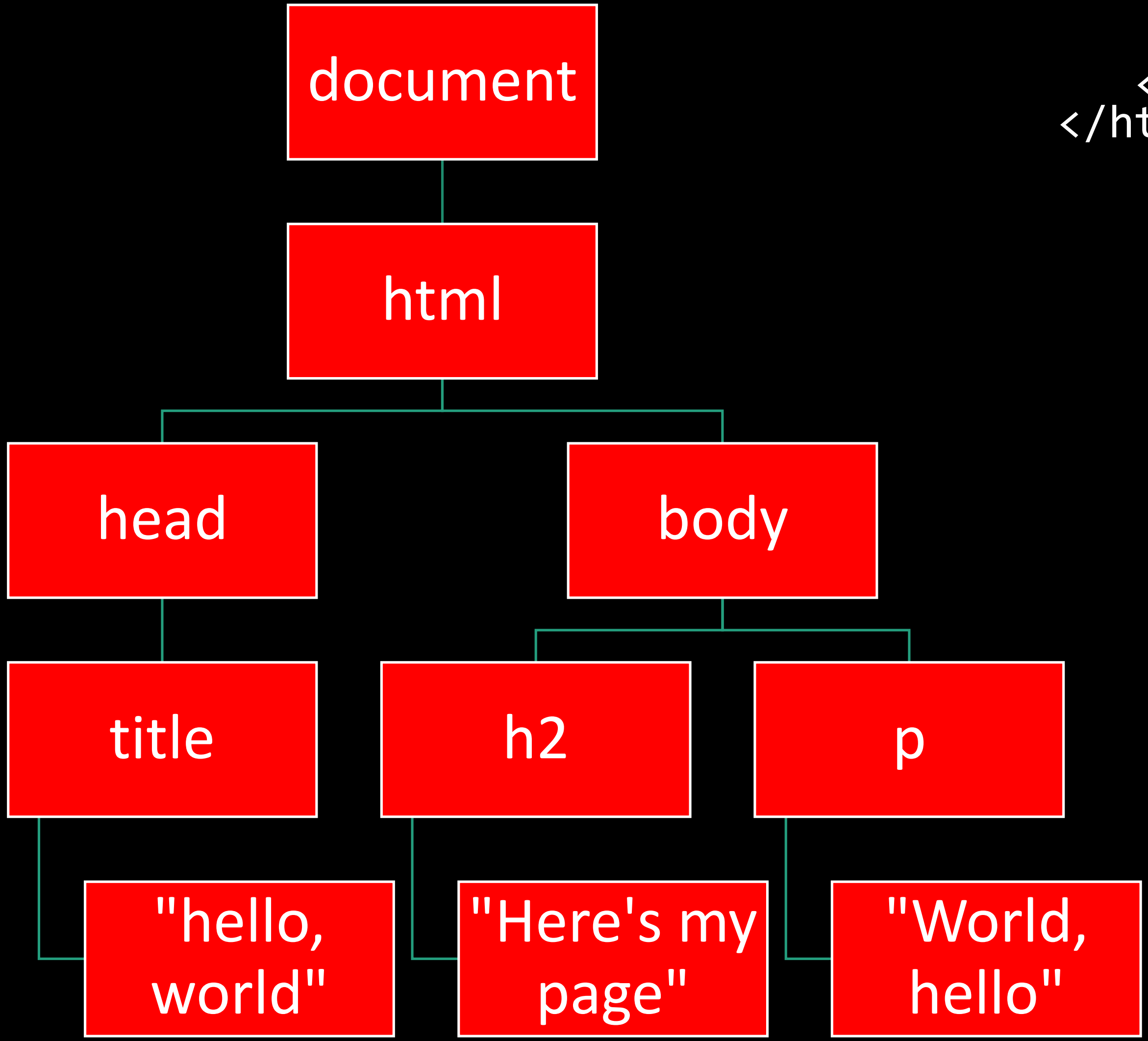
Functions

```
function hello() {
    alert('Hello, world!');
}
```

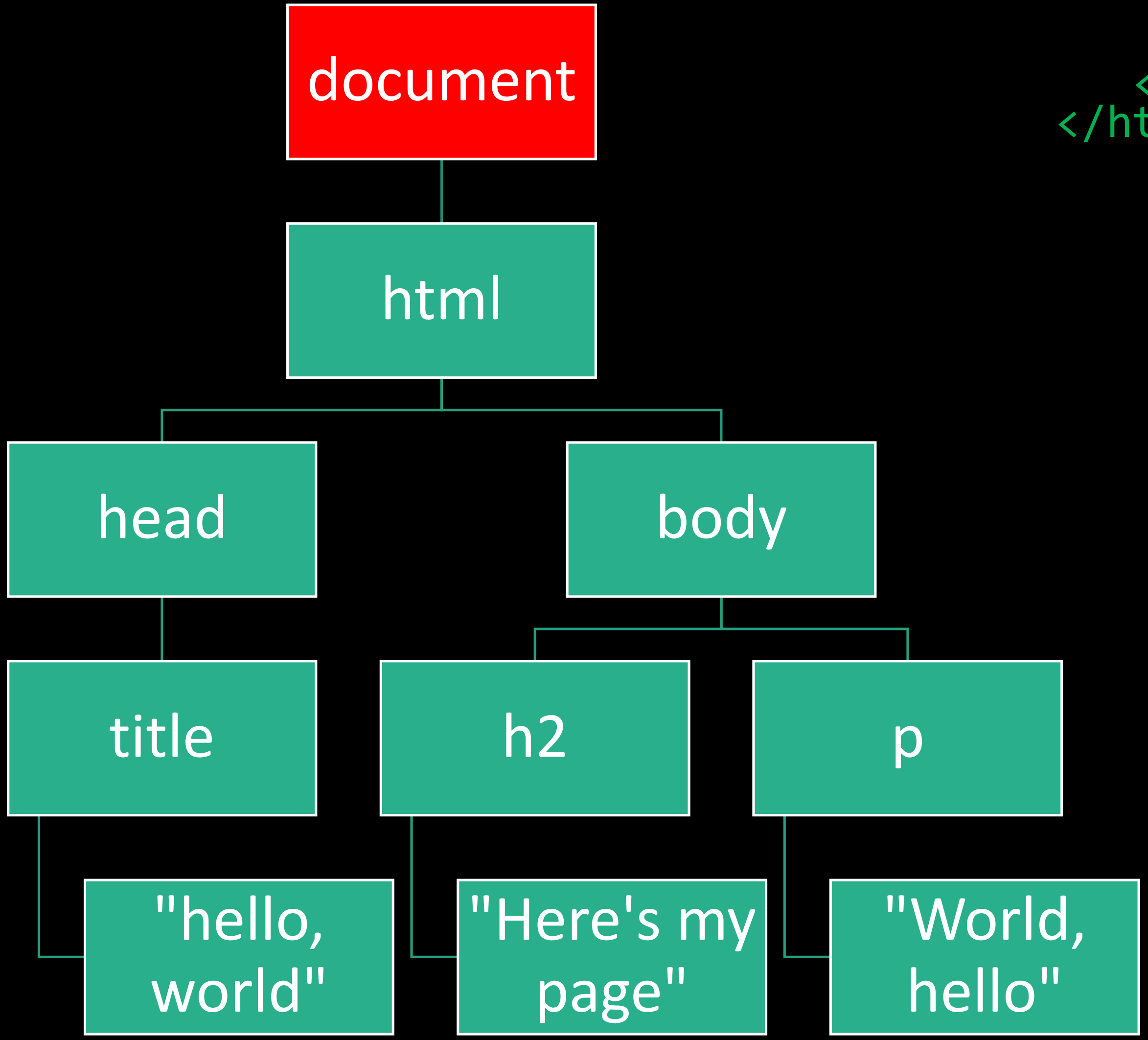
DOM

```
<!DOCTYPE html>
<html>
   <head>
      <title>Hello, world</title>
   </head>
   <body>
      <h2>Here's my page</h2>
      <p>World, hello</p>
   </body>
</html>
```

```html
<!DOCTYPE html>
<html>
   <head>
      <title>Hello, world</title>
   </head>
   <body>
      <h2>Here's my page</h2>
      <p>World, hello</p>
   </body>
</html>
```

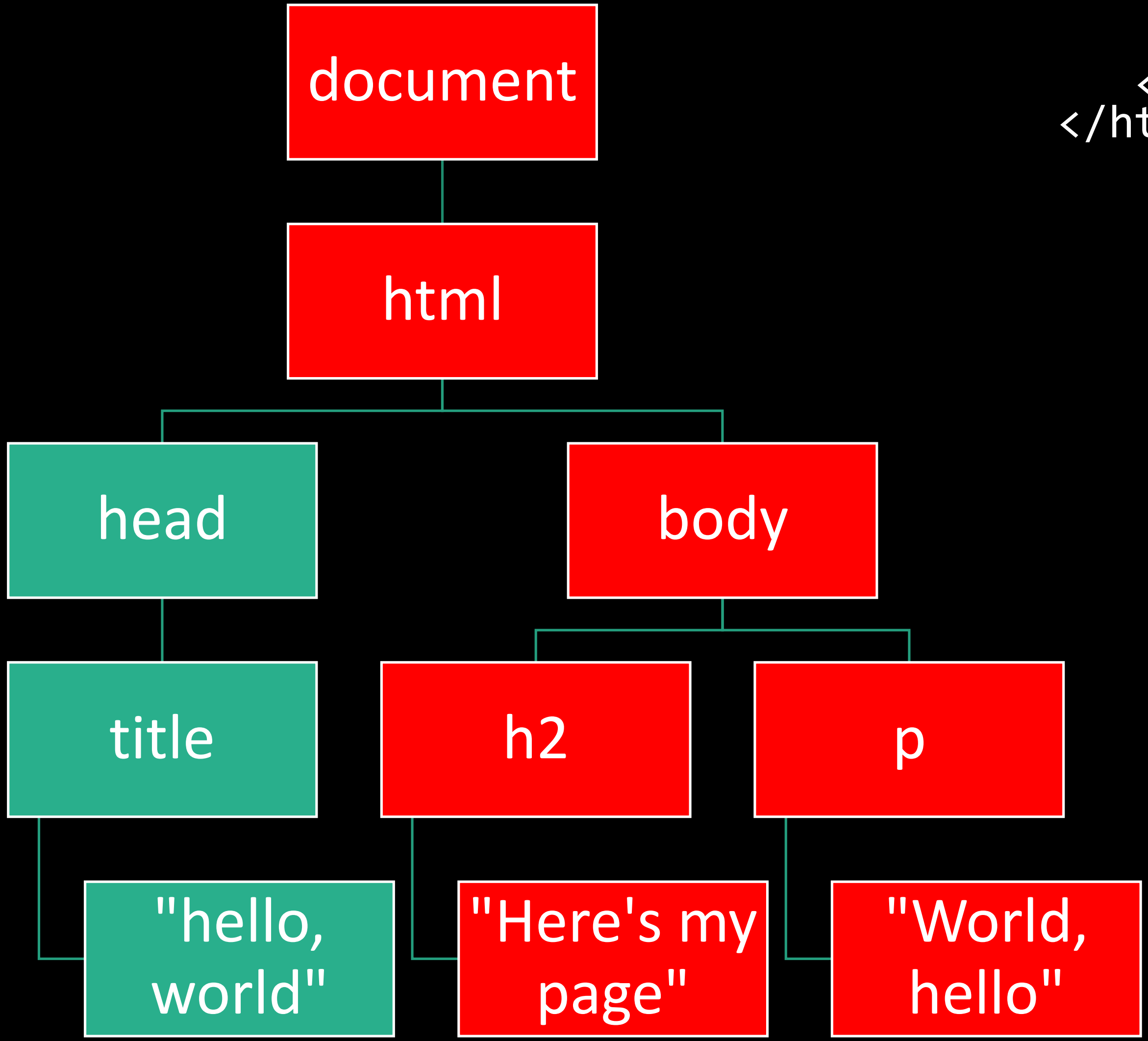


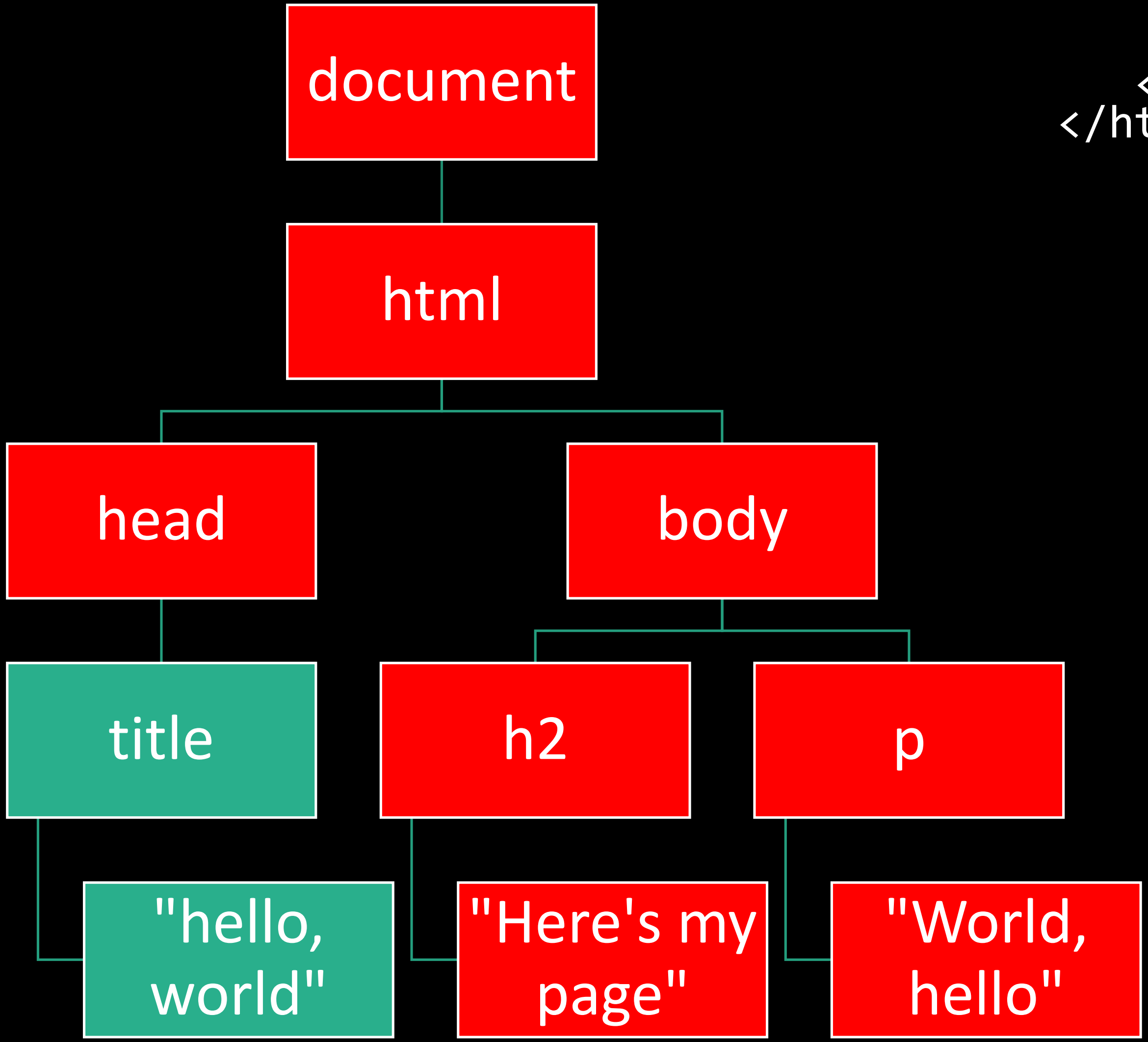
```
<!DOCTYPE html>
<html>
   <head>
      <title>Hello, world</title>
   </head>
   <body>
      <h2>Here's my page</h2>
      <p>World, hello</p>
   </body>
</html>
```



```
<!DOCTYPE html>
<html>
   <head>
      <title>Hello, world</title>
   </head>
   <body>
      <h2>Here's my page</h2>
      <p>World, hello</p>
   </body>
</html>
```

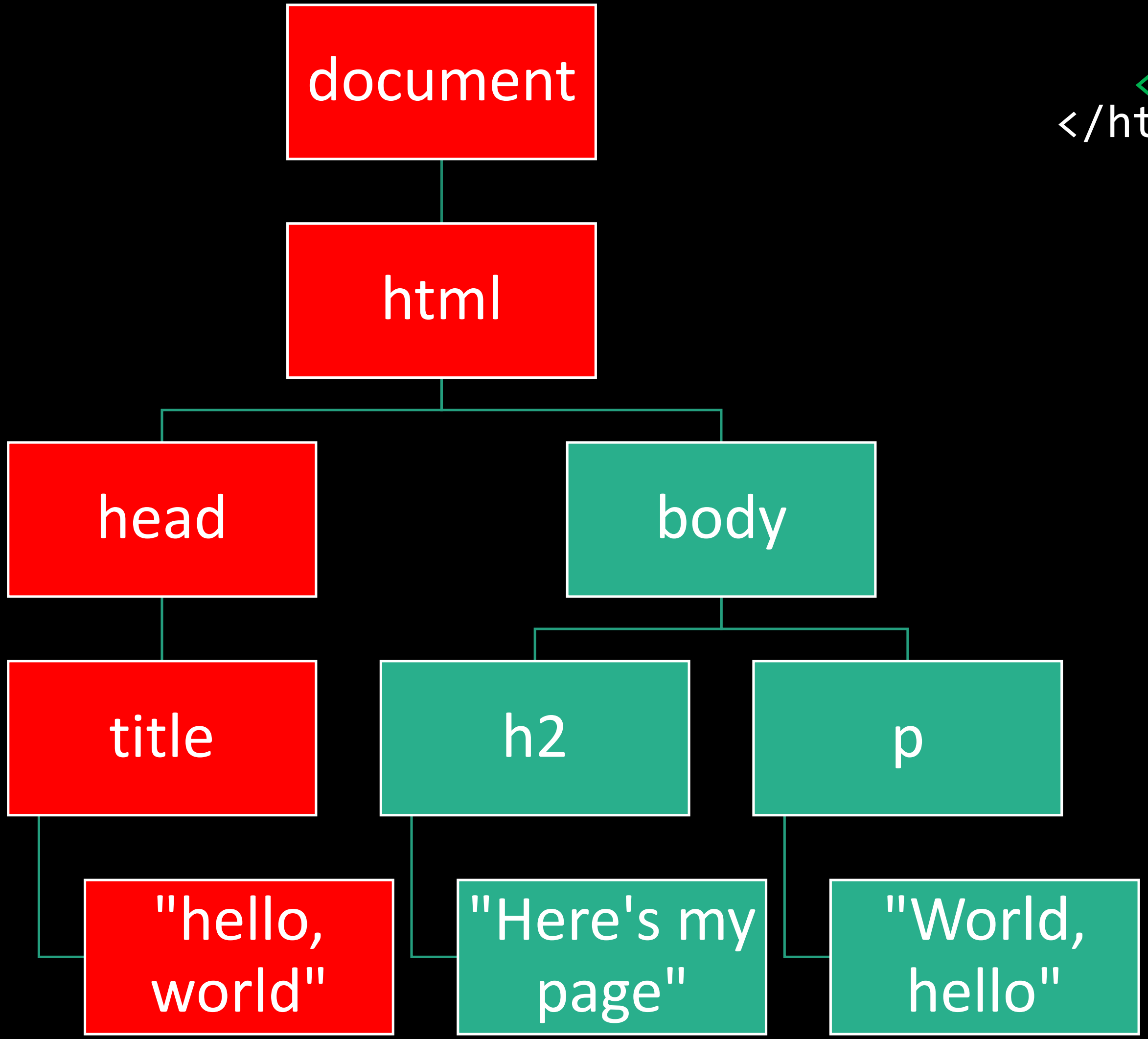
```
<!DOCTYPE html>
<html>
   <head>
      <title>Hello, world</title>
   </head>
   <body>
      <h2>Here's my page</h2>
      <p>World, hello</p>
   </body>
</html>
```





```
<!DOCTYPE html>
<html>
   <head>
      <title>Hello, world</title>
   </head>
   <body>
      <h2>Here's my page</h2>
      <p>World, hello</p>
   </body>
</html>
```

```
<!DOCTYPE html>
<html>
   <head>
      <title>Hello, world</title>
   </head>
   <body>
      <h2>Here's my page</h2>
      <p>World, hello</p>
   </body>
</html>
```



DOM

- The document object is the means by which we can interact with and manipulate web sites using JavaScript.

Query Selector

- `document.querySelector('tag')`
- `document.querySelector('#id')`
- `document.querySelector('.class')`

Conditionals

- `if`
- `else`
- `switch`
- `?:`

Loops

- `while`
- `do ... while`
- `for`
- `for ... in`
- `for ... of`

Template Literals

```
alert(`Counter is at ${counter}!`);
```

Template Literals

```
alert(`Counter is at ${counter}!`);
```

Template Literals

```
alert(`Counter is at ${counter}!`);
```

Template Literals

```
alert(`Counter is at ${counter}!`);
```

Variables

- `const`
- `let`
- `var`

Arrow Functions

```
() => {
    alert('Hello, world!');
}
```

Arrow Functions

```
x => {
   alert(x);
}
```

Arrow Functions

```
x => x * 2
```

Asynchronicity

- In most of the examples we've talked about, the JavaScript code has been running top-to-bottom as it's encountered.

Asynchronicity

- In most of the examples we've talked about, the JavaScript code has been running top-to-bottom as it's encountered.

- Normally this isn't a problem, but it can be a problem if one of the functions we need to execute might take a long time (e.g., a network call).

Asynchronicity

```
const data = fulfillRequest();

console.log(data);
...
```

Asynchronicity

```
const data = fulfillRequest();
 🕐
console.log(data);
...
```

Asynchronicity

```
const data = fulfillRequest();
🕐🕑
console.log(data);
...
```

Asynchronicity

```
const data = fulfillRequest();
🕐🕑🕒
console.log(data);
...
```

Asynchronicity

```
const data = fulfillRequest();
🕐🕑🕒🕓🕔🕕🕖🕗🕘🕙🕚🕛
console.log(data);
...
```

Asynchronicity

```
const data = fulfillRequest();
🕐🕑🕒🕓🕔🕕🕖🕗🕘🕙🕚🕛
console.log(data);
...
```

Asynchronicity

```
fulfillRequest()
.then(data => data.parse())
.then(results => console.log(results))
...
```

Asynchronicity

```
fulfillRequest().then(data => data.parse()).then(results => console.log(results))

...
```

Asynchronicity

```
fulfillRequest()
.then(data => data.parse())
.then(results => console.log(results))
...
```

Asynchronicity

```
fulfillRequest()
.then(data => data.parse())
.then(results => console.log(results))
...
```

If you see a structure like this somewhere, this is indicative of what's known as a JavaScript **promise**, a mechanism for ensuring orderly execution of asynchronous code.

Additional Requests

- Using JavaScript, it is possible for our code to make supplementary HTTP requests without reloading the page.

Additional Requests

- Using JavaScript, it is possible for our code to make supplementary HTTP requests without reloading the page.

- This technique is commonly known as Ajax, and you may have done it before using `XMLHttpRequests`.

Additional Requests

- Using JavaScript, it is possible for our code to make supplementary HTTP requests without reloading the page.
- This technique is commonly known as Ajax, and you may have done it before using `XMLHttpRequests`.
- In ES6, one of the main mechanisms we'll use to achieve this with a promise is `fetch()`.

APIs

API

- **A**pplication **P**rogramming **I**nterfaces are "contracts" of a sort between a client (us) and, usually, a data provider, to give our applications the ability to access data that may be useful to us in some way.

API

- **A**pplication **P**rogramming **I**nterfaces are "contracts" of a sort between a client (us) and, usually, a data provider, to give our applications the ability to access data that may be useful to us in some way.

- In this course, we'll be using APIs from many different service providers and creating projects that leverage data from those providers.

API

- **A**pplication **P**rogramming **I**nterfaces are "contracts" of a sort between a client (us) and, usually, a data provider, to give our applications the ability to access data that may be useful to us in some way.

- In this course, we'll be using APIs from many different service providers and creating projects that leverage data from those providers.

- Learning to parse API docs will be a crucial skill!

Google Books

Lab 1

Lab 1

- Using Harvard Museums' API, create a web application for viewing objects and exhibits.

Lab 1

- Using Harvard Museums' API, create a web application for viewing objects and exhibits.
- Once you plug in your API key, the application will already give you a list of galleries. Your job is to extend the app to show lists of objects within a gallery, and then when you click on an object, show info about that object.

Lab 1

- Using Harvard Museums' API, create a web application for viewing objects and exhibits.
- Once you plug in your API key, the application will already give you a list of galleries. Your job is to extend the app to show lists of objects within a gallery, and then when you click on an object, show info about that object.
- https://github.com/harvardartmuseums/api-docs

Lab 1

- It is allowed to work with one partner on this lab, read the instructions in the specification for the rules for this.

Lab 1

- It is allowed to work with one partner on this lab, read the instructions in the specification for the rules for this.
- Post public questions (i.e., those not containing large code snippets) on Piazza using the **lab1** tag!