

the Master Course

{C0DENATION}

JS & DOM

Introduction

{ CØDENATION }

Learning Objectives

Understand the HTML & DOM structure

To be able to apply changes to the DOM by responding to user interaction



DOM

The DOM

... Document Object Model

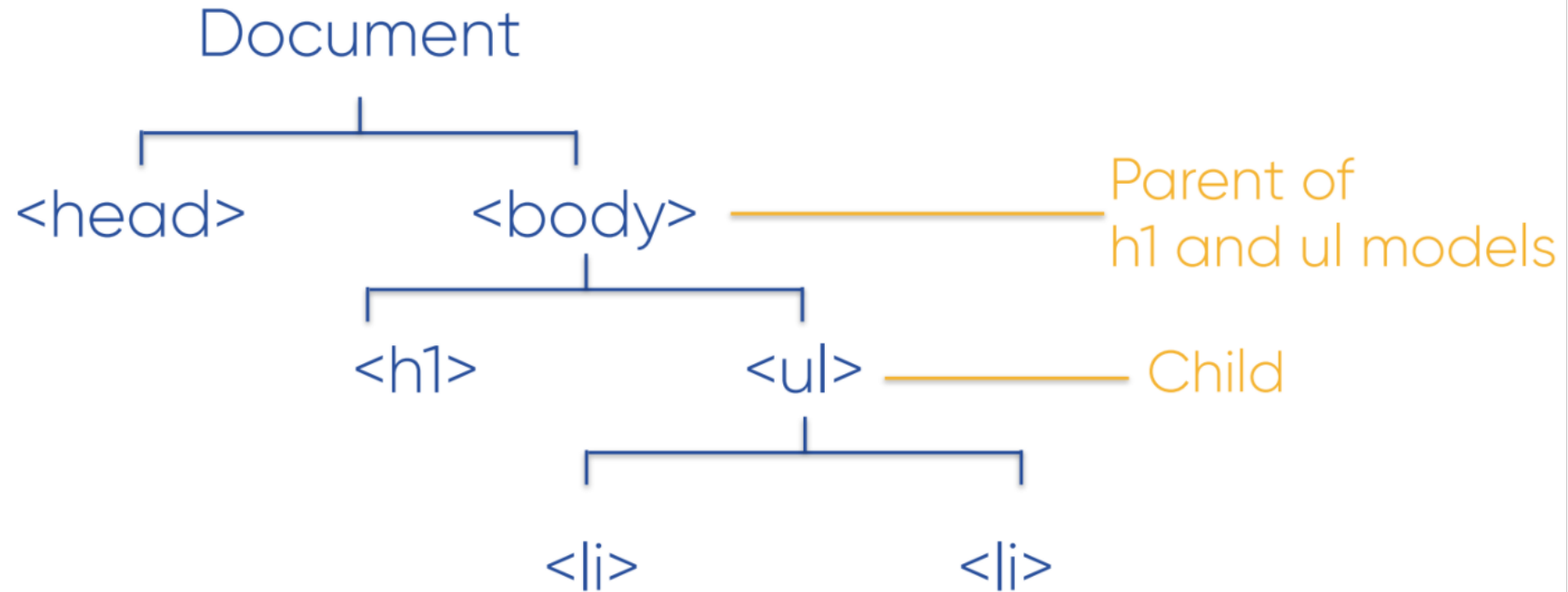
DOM

What is the DOM

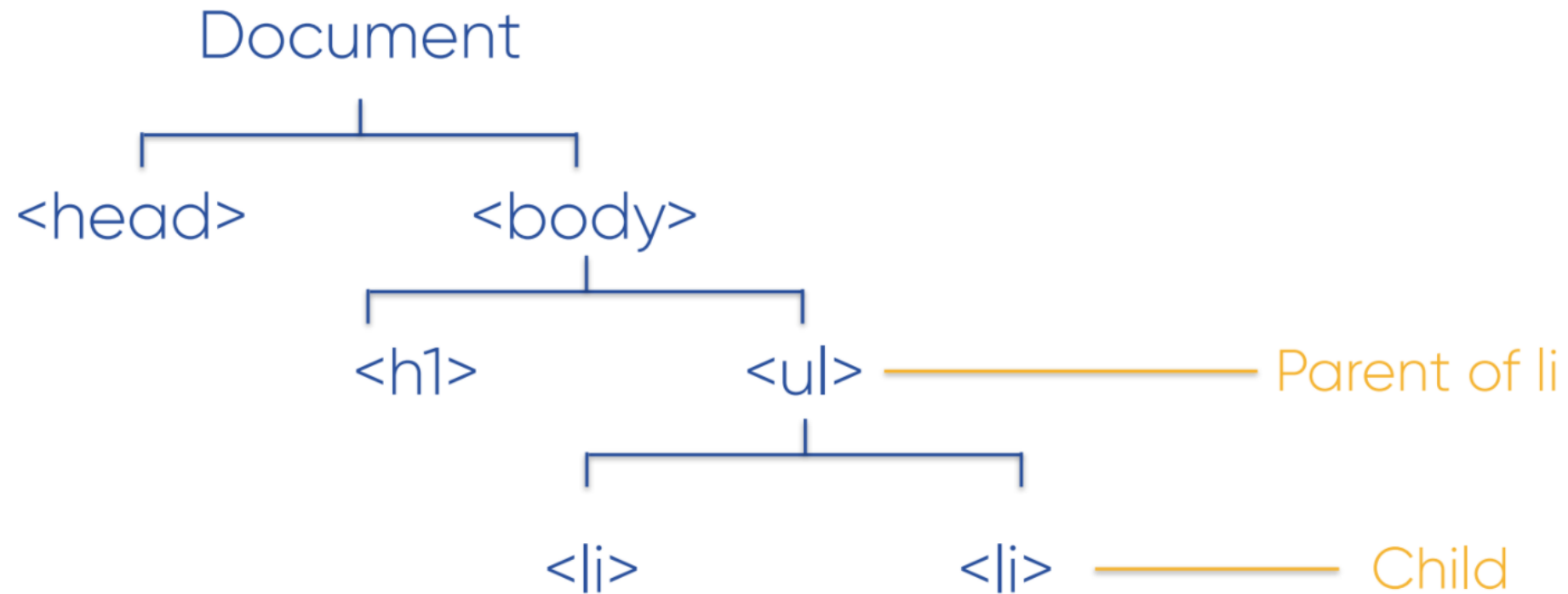
... a **representation of a webpage** that JS can use.
Changes **that JS makes** to the DOM **alter the webpage**.

The DOM represents a webpage as a tree like structure

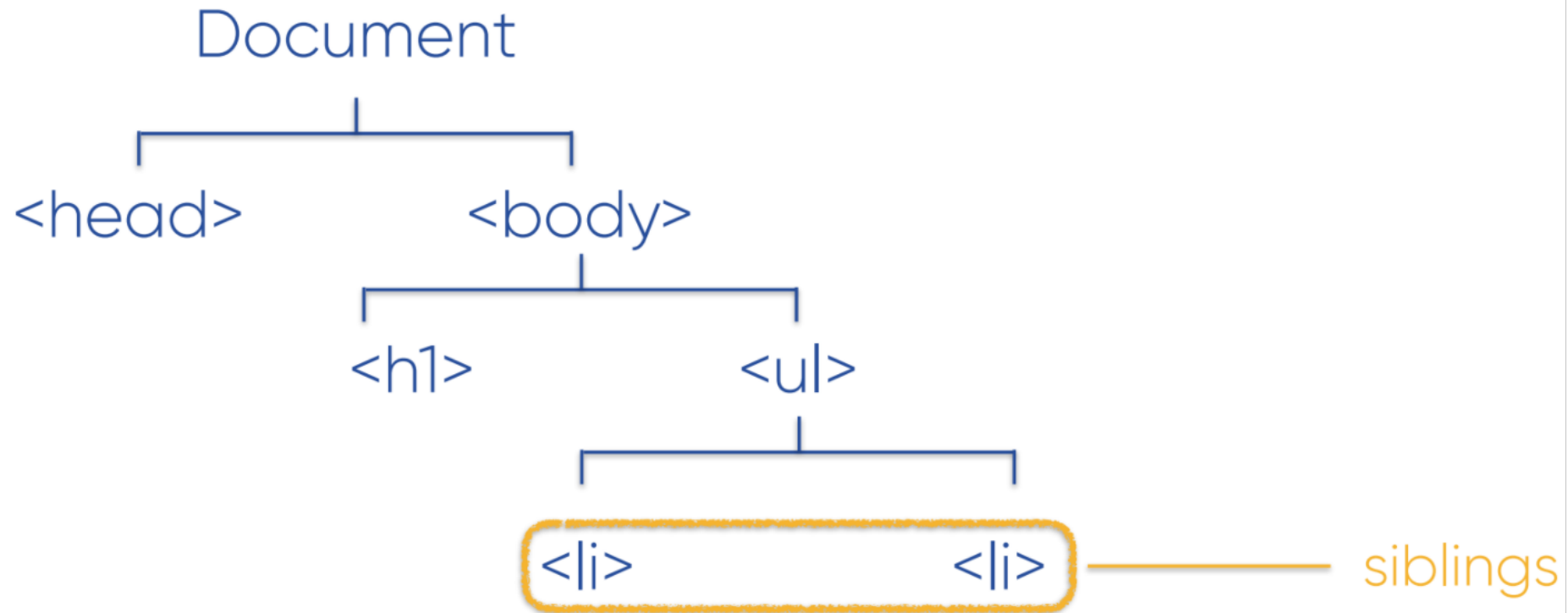
DOM



DOM



DOM



DOM

What can JS do with DOM?

... lots! **Selecting Elements, Reading/Changing
Elements, Responding to User Events**

Connecting JS file

DOM

Inside your HTML file **add a script tag at the bottom of the body** and connect it to your JS file

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Add Event Listener</title>
</head>
<body>
  <h1 id="heading">JS DOM</h1>
  <script src="app.js"></script>
</body>
</html>
```

Why at the bottom?





DOM

Document Object

... document is a **global object** representing
the **HTML and content of the page**



DOM

Document Object

... with JS you can **select and control elements**
of the currently loaded webpage.

e.g Change the heading to a different colour



DOM

Lets look at...

... selecting Elements

Selecting Elements

These are the ways you can select elements but they return different values

Returns Multiple Elements

```
document.getElementsByTagName()  
document.getElementsByClassName()  
document.querySelectorAll()
```

Returns a Single Element

```
document.getElementById()  
document.querySelector()
```

getElementById

DOM

Returns the element with the specified ID. In this case the selected ID is the ul tag

HTML

```
<h1 id="heading">JS DOM</h1>
<ul id="listWrapper">
  <li class="listItem">Item 1</li>
  <li class="listItem">Item 2</li>
  <li class="listItem">Item 3</li>
  <li class="listItem">Item 4</li>
</ul>
```

JS

```
document.getElementById('listWrapper')
```

JS DOM

- Item 1
- Item 2
- Item 3
- Item 4

querySelector

DOM

Returns the first element that matches the specified selector. All examples select the same element

HTML

```
<h1 id="heading">JS DOM</h1>
<ul id="listWrapper">
  <li class="listItem">Item 1</li>
  <li class="listItem">Item 2</li>
  <li class="listItem">Item 3</li>
  <li class="listItem">Item 4</li>
</ul>
```

JS DOM

- Item 1
- Item 2
- Item 3
- Item 4

JS

```
document.querySelector('.listItem')
document.querySelector('#listWrapper li:nth-child(1)')
document.querySelector('li')
```


getElementsByTagNameDOM

Returns a collection of elements with the specified tag name.
Returns a collection even if it only finds 1 element.

HTML

```
<h1 id="heading">JS DOM</h1>
<ul id="listWrapper">
  <li class="listItem">Item 1</li>
  <li class="listItem">Item 2</li>
  <li class="listItem">Item 3</li>
  <li class="listItem">Item 4</li>
</ul>
```

JS

```
document.getElementsByTagName('li')
```

JS DOM

- Item 1
- Item 2
- Item 3
- Item 4

Loop through Elements DOM

We can access all elements in a collection by looping through the collection. We access elements like arrays using square brackets

HTML

```
<h1 id="heading">JS DOM</h1>
<ul id="listWrapper">
  <li class="listItem">Item 1</li>
  <li class="listItem">Item 2</li>
  <li class="listItem">Item 3</li>
  <li class="listItem">Item 4</li>
</ul>
```

JS

```
const listItems = document.getElementsByTagName('li')

for(let i = 0; i < listItems.length; i++){
  listItems[i].style.color = "blue"
}
```

JS DOM

- Item 1
- Item 2
- Item 3
- Item 4

getElementsByClassName

Returns a collection of elements with the specified class name.

HTML

```
<h1 id="heading">JS DOM</h1>
<ul id="listWrapper">
  <li class="listItem">Item 1</li>
  <li class="listItem">Item 2</li>
  <li class="listItem">Item 3</li>
  <li class="listItem">Item 4</li>
</ul>
```

JS

```
document.getElementsByClassName('listItem')[2]
```

JS DOM

- Item 1
- Item 2
- Item 3
- Item 4

querySelectorAll

DOM

Returns a collection of elements with the specified selector

HTML

```
<h1 id="heading">JS DOM</h1>
<ul id="listWrapper">
  <li class="listItem">Item 1</li>
  <li class="listItem">Item 2</li>
  <li class="listItem">Item 3</li>
  <li class="listItem">Item 4</li>
</ul>
```

JS

```
const listItems = document.querySelectorAll('.listItem')

for(let i = 0; i < listItems.length; i++){
  listItems[i].style.color = "blue";
}
```

JS DOM

- Item 1
- Item 2
- Item 3
- Item 4



DOM

Lets look at...

... controlling Elements

textContent

DOM

Sets or returns the text of the specified element, and its descendants

HTML

```
<h1 id="heading">JS DOM</h1>
<ul id="listWrapper">
  <li class="listItem">Item 1</li>
  <li class="listItem">Item 2</li>
  <li class="listItem">Item 3</li>
  <li class="listItem">Item 4</li>
</ul>
```

JS

```
const header = document.getElementById('heading')

header.textContent = 'My List'
```

JS DOM

- Item 1
- Item 2
- Item 3
- Item 4



My List

- Item 1
- Item 2
- Item 3
- Item 4

innerHTML

DOM

Sets or returns the HTML content of the specified element

HTML

```
<h1 id="heading">JS DOM</h1>
<ul id="listWrapper">
  <li class="listItem">Item 1</li>
  <li class="listItem">Item 2</li>
  <li class="listItem">Item 3</li>
  <li class="listItem">Item 4</li>
</ul>
```

JS

```
const list = document.getElementById('listWrapper')
```

```
list.innerHTML = '<li>New List Item</li>'
```

JS DOM

- Item 1
- Item 2
- Item 3
- Item 4

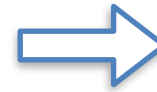


JS DOM

- New List Item

textContent v innerHtml DOM

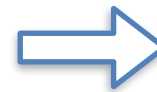
```
const header = document.getElementById('heading')
header.textContent = '<i>Heading</i>'
```



<i>Heading</i>

- Item 1
- Item 2
- Item 3
- Item 4

```
const header = document.getElementById('heading')
header.innerHTML = '<i>Heading</i>'
```



Heading

- Item 1
- Item 2
- Item 3
- Item 4

Styling Elements

DOM

Using `.style` gives you access to all CSS properties. Properties must be typed using camelCase in JS

```
const header = document.getElementById('heading')
const list = document.querySelector('ul')
const listItems = document.getElementsByClassName('listItem')

header.style.border = '3px solid green'
list.style.backgroundColor = 'yellow';
listItems[1].style.display= 'none';
```

JS DOM

- Item 1
- Item 3
- Item 4



DOM

Lets look at...

... adding Event Listeners

Add Event Listener

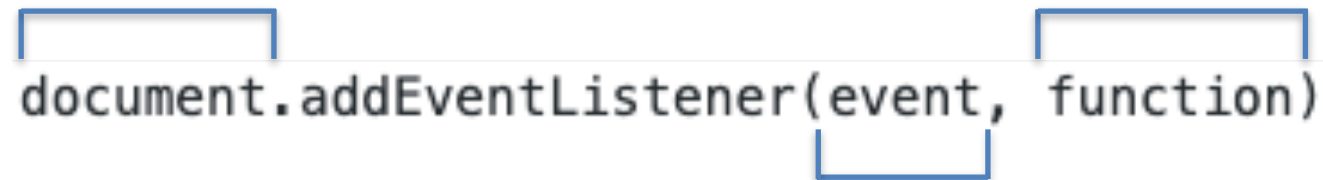
DOM

This method adds a function that will be called when the specified event is triggered.

Element the event is added to
(document adds it to whole page)

Function to run when event is triggered

```
document.addEventListener(event, function)
```



Event to listen for

click Event Example

DOM

Adds a click event to the entire document. The function will get called when someone clicks anywhere on the document

HTML

```
<h1 id="heading">JS DOM</h1>
<ul id="listWrapper">
  <li class="listItem">Item 1</li>
  <li class="listItem">Item 2</li>
  <li class="listItem">Item 3</li>
  <li class="listItem">Item 4</li>
</ul>
```

JS

```
const header = document.getElementById('heading')

document.addEventListener('click', () => {
  header.textContent = "Its Changed!"
})
```

JS DOM

- Item 1
- Item 2
- Item 3
- Item 4

After click on page



Its Changed!

- Item 1
- Item 2
- Item 3
- Item 4

{ C⓪DENATION }

mouseover Event ExampleDOM

This method adds a function that will be called when the specified event is triggered.

HTML

```
<h1 id="heading">My Heading</h1>
```

JS

```
const heading = document.getElementById('heading')

heading.addEventListener('mouseover', () => {
  heading.textContent = heading.textContent.toUpperCase()
})

heading.addEventListener('mouseout', () => {
  heading.textContent = heading.textContent.toLowerCase()
})
```

my heading

When hovering
over heading



MY HEADING

Events

DOM

There are many events that we can apply to the document or specific elements

mousedown

mouseover

keydown

click

keypress

mouseup

keyup

mousemove

<https://developer.mozilla.org/en-US/docs/Web/API/Element#events>

{ CØDENATION }

Input Values

DOM

You can get the text typed in an input box by using the value property

HTML

```
<h1 id="heading">JS DOM</h1>
<input type="text" id="inputBox">
<button id="submit">Submit</button>
```

JS

```
const heading = document.getElementById('heading')
const input = document.getElementById('inputBox')
const submitBtn = document.getElementById('submit')

submitBtn.addEventListener('click', () => {
  heading.textContent = input.value
})
```

JS DOM

On button click 

Hello There

Changing Attributes

Sets or returns the text of the specified element, and its descendants

HTML

```
<h1 id="heading">My Picture</h1>  

```

JS

```
const image = document.getElementById('myImage')  
  
image.src = 'https://rippلز.co.uk/static/  
code-nation-logo-6b1b26f34aa37a48ac086ab2340456a2.png'
```

DOM

My Picture



My Picture





DOM

Lets look at...

... the Event Object

Event Object

DOM

An object that holds information about event that's been triggered.
event.target references the element the event was added to

```
document.addEventListener('click', (event) => {  
  console.log(event)  
  console.log(event.target)  
})
```

```
app.js:3  
▶ PointerEvent {isTrusted: true, pointerId: 1,  
  width: 1, height: 1, pressure: 0, ...}  
app.js:4  
<html lang="en">  
  ▶ <head>...</head>  
  ▶ <body>...</body>  
</html>
```

Learning Objectives

Understand the HTML & DOM structure

To be able to apply changes to the DOM by responding to user interaction

Activity 1

Create an image and a button in your HTML. Make a toggle so that when you click the button, hide the image. Then when you click the button again the image is visible

Activity 2

Create an image, button, and input in your HTML. Change the source of the image to whatever image url is inside the input box when the button is clicked.

Activity 3

Create a heading, button, and input in your HTML. Change the text colour of the heading to whatever colour is typed in the input when the button is clicked

Activity 4

Using the event object, get the x and y coordinates of where you click on the page and display the values in a p tag.

Hint: look into clientX and clientY