前端網絡開發人員課程
(二) 進階網絡程式設計

1. JS DOM I: Elements I

Presented by Krystal Institute

Learning Objective

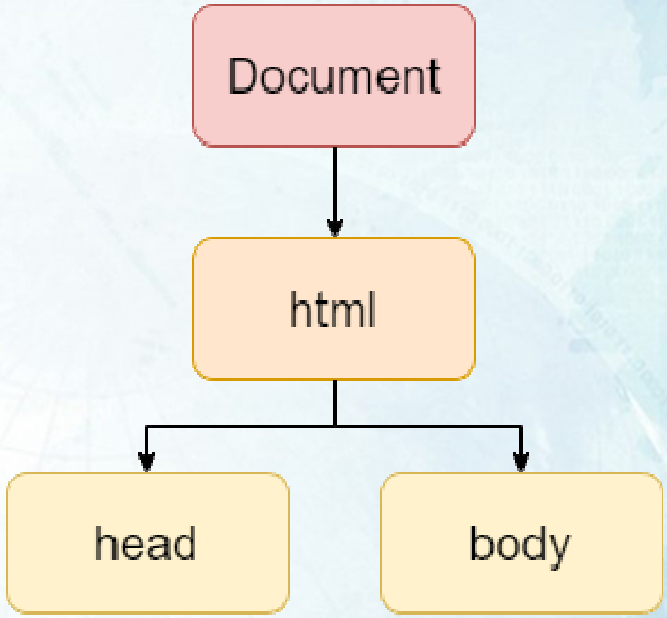
- Understand what DOM is, and its relationships with HTML
- Learn how to select elements with JavaScript

Content

1.1 Introduction to HTML DOM

Document Object Model (DOM)

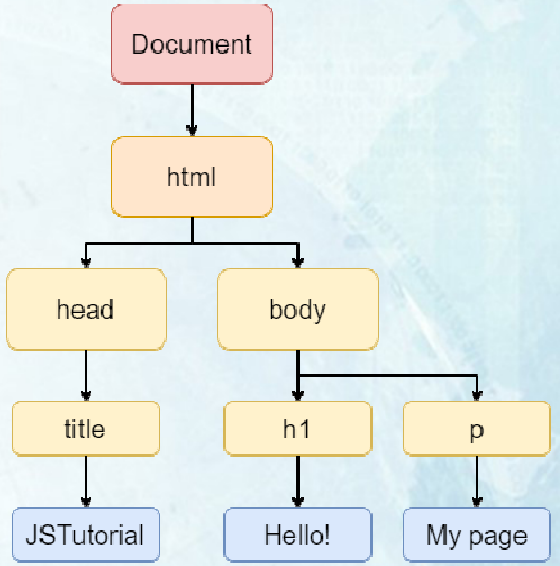
- API for manipulating HTML and XML documents
- Defines the structure of the document in a tree of nodes
- Allows for addition, removal, and modification of nodes



Document Object Model (DOM)

- Document is the root node
- <html> is the document element
- This is how DOM (right) looks like with html code (left)

```
<html>
<body>
    <h1>Hello!</h1>
    <p>My page</p>
</body>
</html>
```



Nodes

- It is a abstract concept that many HTML API used
- Everything in DOM is a node of various types
- There are mainly three types of nodes:
  - Element Nodes
  - Text Nodes
  - Comment Nodes

Element Nodes

- Forms the tree structure
- <html> is the root element
- <head> and <body> is the children of <html>
- Can add **classes, attributes, styles**, and interact with them using JavaScript
- Example of element node

```
<div class="wrapper main" style="color: red">
</div>
```

Text Nodes

- Text inside elements are text nodes with exceptions
- Contains newline and spaces

Nodes: Question

What node comes after in the code below?

```
<div id="t1">
    <p>A paragraph</p>
</div>
```

Nodes: Answer

What node comes after in the code on right?

```
<div id="t1">
    <p>A paragraph</p>
</div>
```

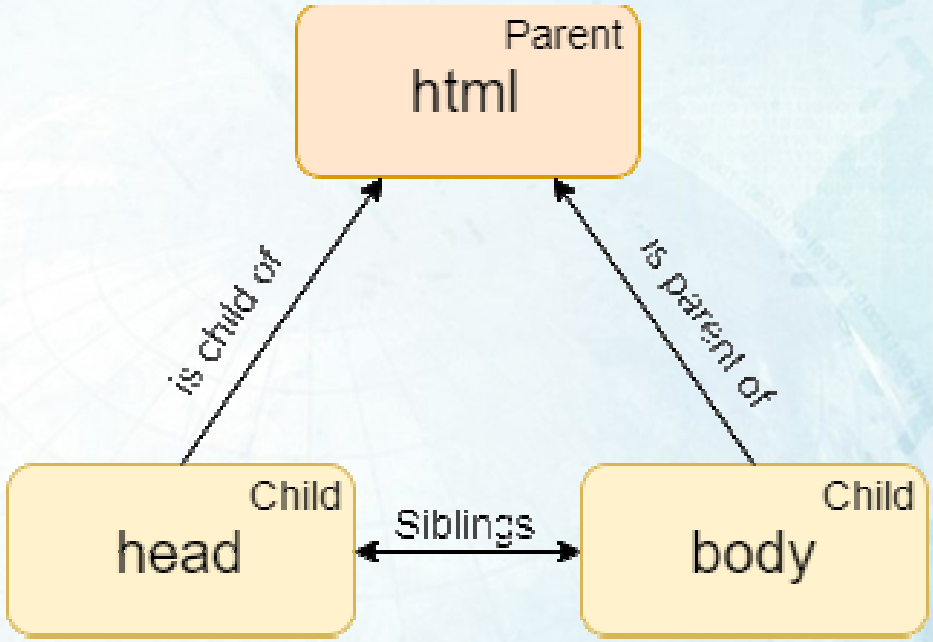
Answer: Text Node

Explanation: has indentations and spaces before <p>

Indentations and spaces counts as text nodes

Node Relationships

- Relationships between each node are the same as **a traditional family tree**
- <body> is a child node of <html>, and <html> is parent of <body>
- <body> and <head> are siblings, as they both have the same parent <html>



1.2 DOM: Selecting Elements

Selecting by id

- A HTML Element often contains an id attribute

```
<div id="main"></div>
```

- id is unique, and is used to identify specific elements in the document
- BUT the same id on multiple elements can exist (it can, but it shouldn't)
- id are case-sensitive

Selecting by id: Activity

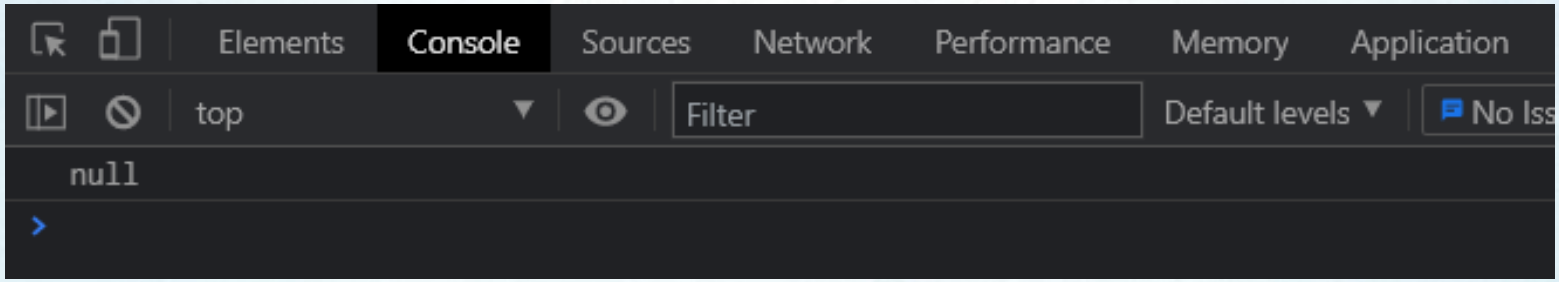
- Activity: display a paragraph element
- Create an element with id attribute
- Select an element by using getElementById
- Display with console.log
- Reminder: press F12 in web browser to open Developer Tools

```
<html>
<head>
    <title>JSTutorial</title>
</head>
<body>
    <p id="paragraph">Text here</p>
<script>
    const p =
document.getElementById("paragraph")
    console.log(p)
</script>
</body>
</html>
```

Selecting by id

- If no id exist when selecting one, null will return

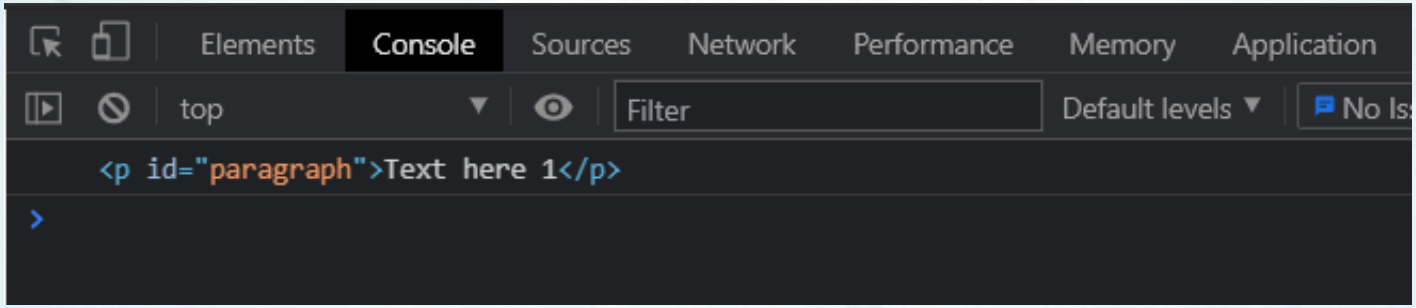
```
<body>
    <p id="p">Text here</p>
<script>
    const p = document.getElementById("paragraph")
    console.log(p)
</script>
</body>
```



Selecting by id

- If multiple elements share the same id, getElementById() returns the first element it encounters

```
<body>
    <p id="paragraph">Text here 1</p>
    <p id="paragraph">Text here 2</p>
    <p id="paragraph">Text here 3</p>
<script>
    const p = document.getElementById("paragraph")
    console.log(p)
</script>
</body>
```



Selecting by Name

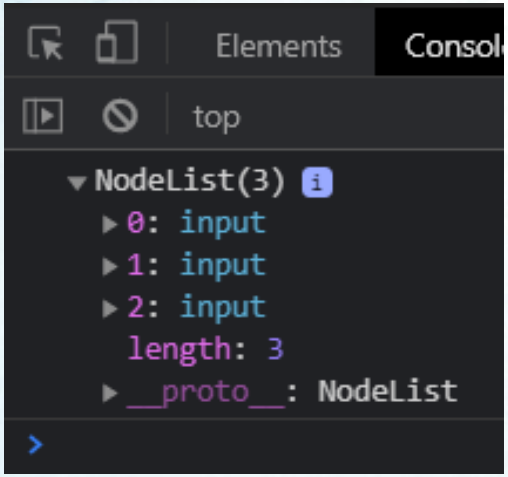
- Every element may have a name attribute
- Multiple elements can share the same name

```
<input type="text" name="textinput">
<input type="text" name="textinput">
<input type="text" name="textinput">
```

Selecting by Name

- Using GetElementsByName returns a array of elements with the specified name

```
<body>
    <input type="text" name="textinput">
    <input type="text" name="textinput">
    <input type="text" name="textinput">
<script>
    let p = document.getElementsByName("textinput")
    console.log(p)
</script>
</boddy>
```



Selecting by Name

- The collection of elements is live (it will update along with the DOM Tree)
- Additions and removals of elements with the same name will be automatically updated

Selecting by Name: Activity

- Activity: display full name from 2 inputs
- Create 2 input elements: firstname and lastname with name="name"
- Create a simple button with onclick function
- Get name and value of elements using GetElementsByName
- Display full name with console.log

```
<body>
    First Name: <input type="text" name="name">
    Last Name: <input type="text" name="name">
    <button type="button" onclick="getname()">Get Name</button>
<script>
    function getname() {
        const p = document.getElementsByName("name");
        let fullname = "";
        for (let i = 0; i < p.length; i++) {
            fullname += p[i].value;
            fullname += " ";
        };
        console.log(fullname);
    };
</script>
</body>
```

Selecting by Tag Name

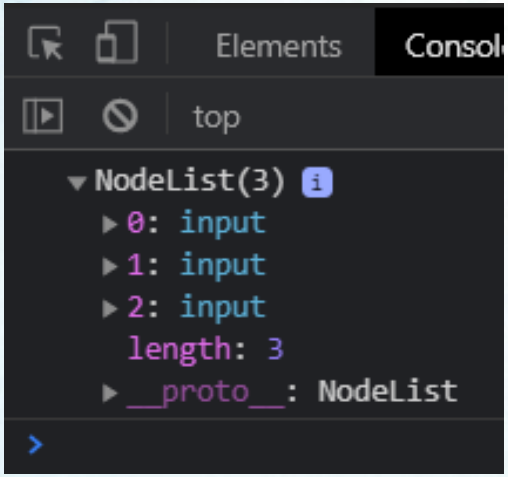
- Tags are element nodes mentioned before
- Text encased inside <> is a tag, it determines what type of element it will be
- There is 1 <body> tag, 1 <h1> tag and 3 <h2> tags in the code on right

```
<body>
    <h1>Drinks</h1>
    <h2>Milk</h2>
    <h2>Tea</h2>
    <h2>Soda</h2>
</body>
```

Selecting by Tag Name

- Returns an array
- Is live
- Similar to getElementsByName

```
<body>
    <input type="text" name="textinput">
    <input type="text" name="textinput">
    <input type="text" name="textinput">
<script>
    let p = document.getElementsByTagName("input")
    console.log(p)
</script>
</body>
```



Selecting by Tag Name: Exercise

- Try to create a website with a button that displays the number of <h2> tags and a button that displays the number of <p> tags

- Use at least 2 <h2> tags and 2 <p> tags

```
<h2>First heading</h2>
<p>This is the first paragraph.</p>
<h2>Second heading</h2>
<p>This is the second paragraph.</p>
<h2>Third heading</h2>
<p>This is the third paragraph.</p>
```

Selecting by Tag Name: Solution

- Using getElementsByTagName, the length of the tags can be assigned and displayed

```
<body>
    <h1>Title</h1>
    <h2>First heading</h2>
    <p>This is the first paragraph.</p>
    <h2>Second heading</h2>
    <p>This is the second paragraph.</p>
    <h2>Third heading</h2>
    <p>This is the third paragraph.</p>

    <button type="button" onclick="count('h2')">Count h2</button>
    <button type="button" onclick="count('p')">Count p</button>

    <script>
        function count(key) {
           console.log(document.getElementsByTagName(key).length)
        }
    </script>
</body>
```

Selecting by Class Name

- The class attribute contains space-separated list of classes

```
<button class="btn btn-primary">Click me</button>
```

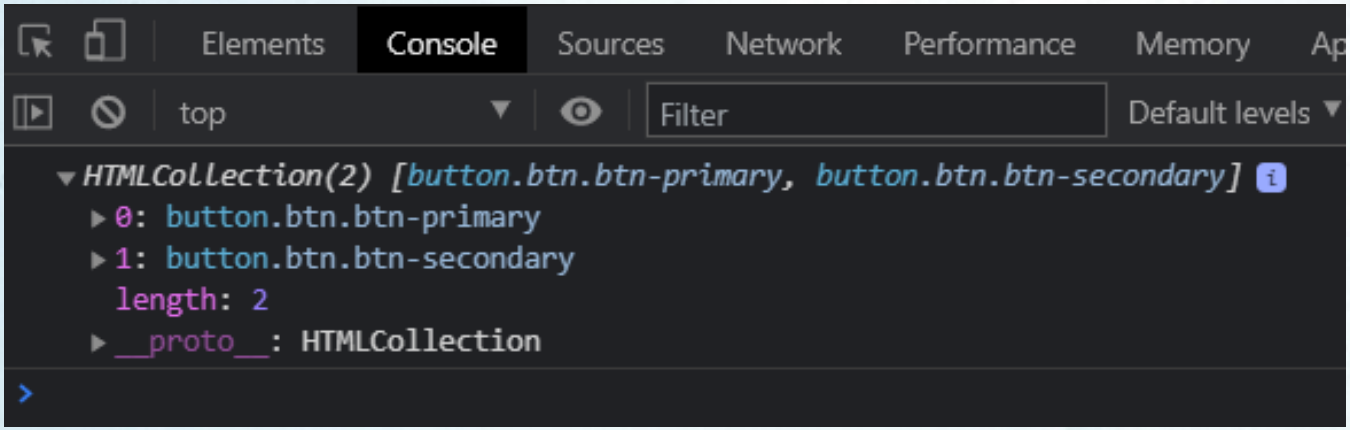
- Case-sensitive

- The button element on the right has btn and btn-primary classes

Selecting by Class Name

- Using getElementsByClassName will return an array, similar to getelementsbyName and TagName

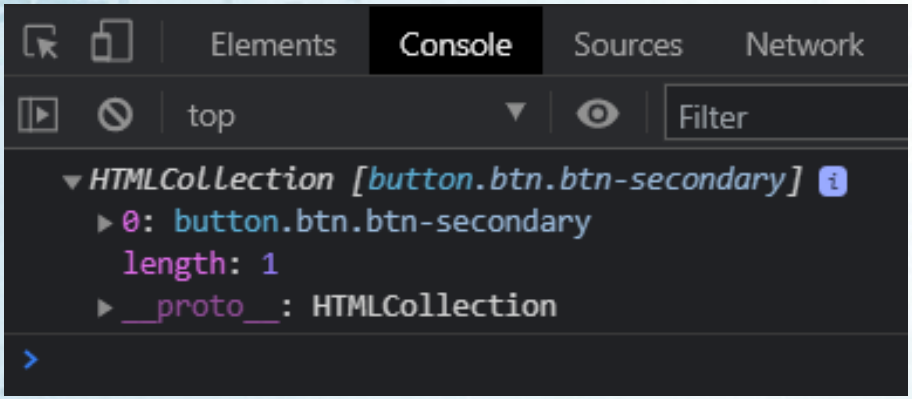
```
<button class="btn btn-primary"></button>
<button class="btn btn-secondary"></button>
<button class="submit"></button>
<script>
    let btn = document.getElementsByClassName("btn")
    console.log(btn)
</script>
```



Selecting by Class Name

- Use white space to separate each class to match elements by multiple classes

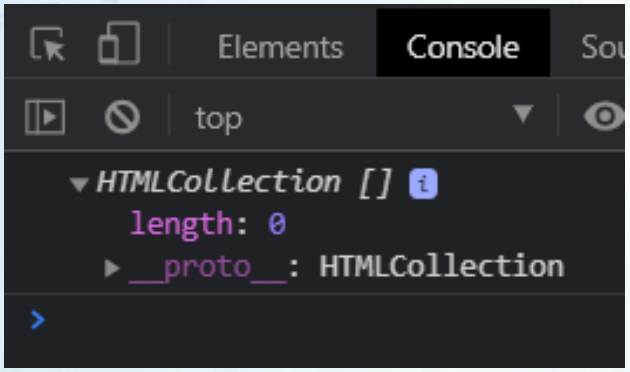
```
<button class="btn btn-primary"></button>
<button class="btn btn-secondary"></button>
<button class="submit"></button>
<script>
    let btn = document.getElementsByClassName("btn
btn-secondary")
    console.log(btn)
</script>
```



Selecting by Class Name

- Using Class selectors will not work in GetElementsByClassName

```
<button class="btn btn-primary"></button>
<button class="btn btn-secondary"></button>
<button class="submit"></button>
<script>
    let btn = document.getElementsByClassName(".btn")
    console.log(btn)
</script>
```



Selecting by Class Name: Question

- What will the code below display in the console?

```
<body>
    <ul>
        <li class="item">JS</li>
        <li class="item">HTML</li>
        <li class="item active highlight">CSS</li>
        <li class="item">DOM</li>
        <li class="active item highlight">Python</li>
    </ul>
<script>
    let item = document.getElementsByClassName("active item highlight")
    console.log(item.length)
</script>
</body>
```

Selecting by Class Name: Answer

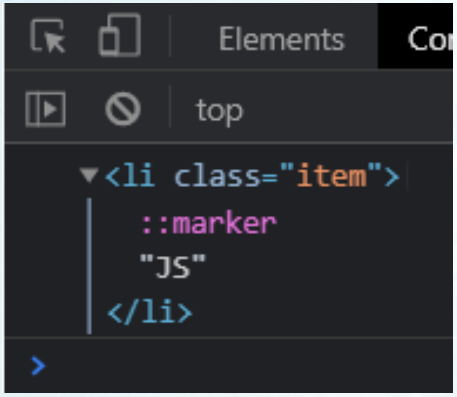
- What will the code below display in the console?

- Answer: 2
- Order doesn't affect selection of elements by multiple classes

```
<body>
    <ul>
        <li class="item">JS</li>
        <li class="item">HTML</li>
        <li class="item active highlight">CSS</li>
        <li class="item">DOM</li>
        <li class="active item highlight">Python</li>
    </ul>
<script>
    let item = document.getElementsByClassName("active item highlight")
    console.log(item.length)
</script>
</body>
```

Query Selector

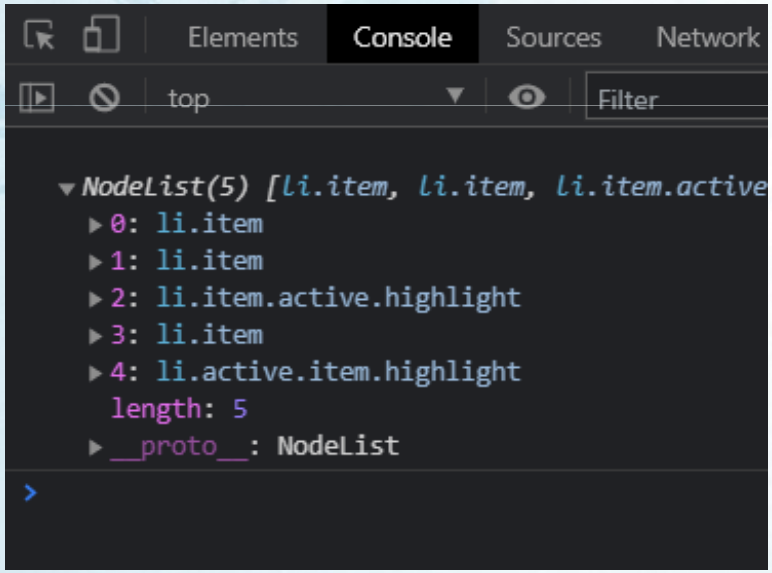
- querySelector uses one or more CSS selectors (e.g. div, li) to find matching elements




```
    <ul>
        <li class="item">JS</li>
        <li class="item">HTML</li>
        <li class="item active highlight">CSS</li>
        <li class="item">DOM</li>
        <li class="active item highlight">Python</li>
    </ul>
<script>
    let item = document.querySelector("li")
    console.log(item)
</script>
```

Query Selector All

- querySelectorAll returns a static NodeList that match the CSS Selector

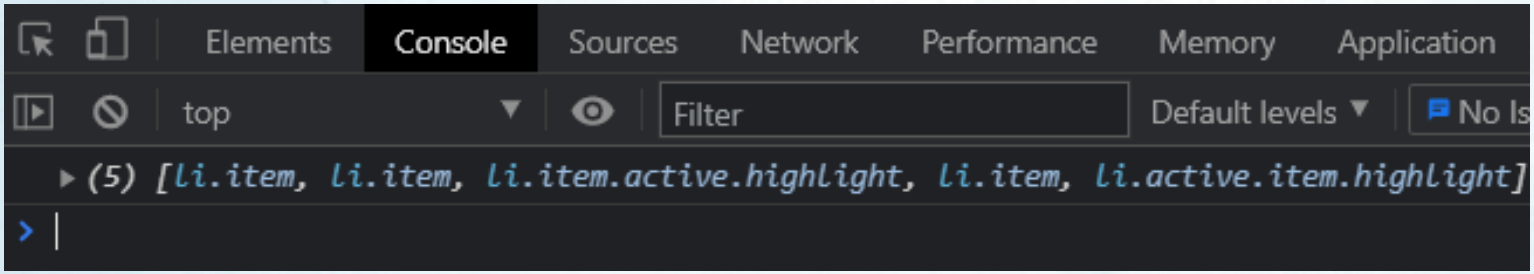



```
    <ul>
        <li class="item">JS</li>
        <li class="item">HTML</li>
        <li class="item active highlight">CSS</li>
        <li class="item">DOM</li>
        <li class="active item highlight">Python</li>
    </ul>
<script>
    let item = document.querySelectorAll("li")
    console.log(item)
</script>
```

Query Selector All

- Note that NodeList is static (will not respond to dynamic changes)
- To convert it into an array, Array.form() should be used

```
<script>
    let item = Array.from(document.querySelectorAll("li"));
    console.log(item)
</script>
```



Query Selector: Example

- On the right is a website that describes keyboards and mouse for the public

```
<html>
<head>
    <title>JSTutorial</title>
</head>
<body>
    <div class="main">
        <h1>Welcome to Online Store!</h1>
        <h3 id="heading1">Keyboards</h3>
        <p data-mark="10"><br><br>Keyboards are
essential to the operation of a computer...</p>
        <h3 id="heading2">Mouse</h3>
        <p data-mark="30"><br><br>Mouse is another
important tool on desktop navigation...</p>
    </div>
</body>
</html>
```

Query Selector: Universal Selector

- A * denotes a universal selector, it matches elements of any type
- Using querySelector will return the first element (<html> and everything inside <html>) found in the document

```
<html>
<head>
    <title>JSTutorial</title>
</head>
<body>
    <div class="main">
        <h1>Welcome to Online Store!</h1>
        <h3 id="heading1">Keyboards</h3>
        <p data-mark="10"><br><br>Keyboards are
essential to the operation of a computer...</p>
        <h3 id="heading2">Mouse</h3>
        <p data-mark="30"><br><br>Mouse is another
important tool on desktop navigation...</p>
    </div>
</body>
</html>
```

Query Selector: Universal Selector

- Using querySelectorAll will return a Nodelist of all elements found in the document

```
<html>
<head>
    <title>JSTutorial</title>
</head>
<body>
    <div class="main">
        <h1>Welcome to Online Store!</h1>
        <h3 id="heading1">Keyboards</h3>
        <p data-mark="10"><br><br>Keyboards are
essential to the operation of a computer...</p>
        <h3 id="heading2">Mouse</h3>
        <p data-mark="30"><br><br>Mouse is another
important tool on desktop navigation...</p>
    </div>
</body>
</html>
```

Query Selector: Type Selector

- Using Type Selector (e.g. h1, div) will return matching element(s) with the correct tags
- querySelectorAll will return a NodeList instead.

```
<script>
    let p = document.querySelector("h3")
    console.log(p)
</script>
```

```
<script>
    let p = document.querySelectorAll(“h3")
    console.log(p)
</script>
```

Query Selector: Class Selector

- Using Class selectors (e.g. .main) will return matching element(s)

```
<script>
    let p = document.querySelector(".main")
    console.log(p)
</script>
```

- querySelectorAll will return a NodeList instead.

```
<script>
    let p = document.querySelectorAll(".main")
    console.log(p)
</script>
```

Query Selector: Id Selector

- Using Id selectors (e.g. #heading1) will return matching element(s)

```
<script>
    let p = document.querySelector("#heading1")
    console.log(p)
</script>
```

- querySelectorAll will return a NodeList instead.

```
<script>
    let p = document.querySelectorAll("#heading1")
    console.log(p)
</script>
```

Query Selector: Attribute Selector

- Using [Attribute] will return elements with matching attributes
- [Attribute=value] can be used to find elements with attributes of specific value

```
<script>
    let p = document.querySelector("[data-mark='10']")
    console.log(p)
</script>
```

```
<script>
    let p = document.querySelectorAll("[data-mark]")
    console.log(p)
</script>
```

Query Selector: Grouping Selectors

- Use a comma to group multiple selectors
- querySelector will return element(s) matching ANY of the selectors

```
<script>
    let p = document.querySelector(".main, #heading1")
    console.log(p)
</script>
```

```
<script>
    let p = document.querySelectorAll(".main, #heading1")
    console.log(p)
</script>
```

Query Selector: Combinators

- Combinators are used in querySelector to find elements with conditions

- Using a space between 2 selectors to find elements inside another (e.g. matching all <p> inside)

```
<div>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
  <section><p>Paragraph 3</p></section>
</div>

<p>Paragraph 4</p>
<p>Paragraph 5</p>
```

```
document.querySelectorAll("div p")
```

Query Selector: Combinators

- Using a > will match elements that are directly inside of another (e.g.
 is not directly inside , <p> is)

```
<div>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
  <section><p>Paragraph 3</p></section>
  <p>Paragraph 4</p>
</div>
```

```
document.querySelectorAll("div>p")
```

Query Selector: Combinators

- Using a ~ will match first/all elements that follows one another (siblings) (e.g. <p> in paragraph 4 and 5 follows , but not <p> in paragraph 1)

```
<p>Paragraph 1</p>

<div>
  <p>Paragraph 2<p>
  <p>Paragraph 3<p>
</div>

<p>Paragraph 4.</p>
<p>Paragraph 5.</p>
```

```
document.querySelectorAll("div~p")
```

Query Selector: Combinators

- Using a + will match elements that directly follows one another (e.g. <p> in paragraph 3 follows)

```
<div>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
</div>

<p>Paragraph 3</p>
<p>Paragraph 4</p>
```

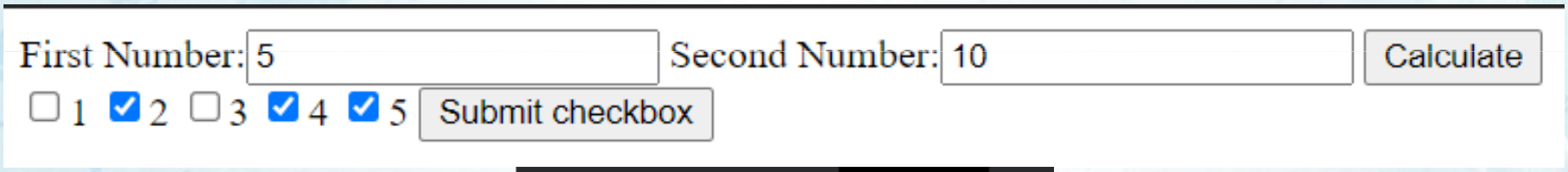
```
document.querySelector("div+p")
```

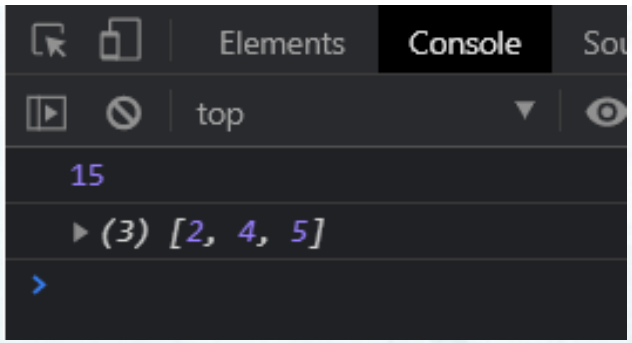
Exercise

- Create a website that…
- Has 2 input boxes for 2 numbers that can be summed and displayed in the console (using GetElementById)
- Has 5 check boxes with values 1 to 5 that displays a list of each checked number (using GetElementByName)
- Please complete it by the end of the lesson

Exercise example

- Inputs: 5, 10 2, 4, 5
- Outputs: 15 [2, 4, 5]





References

- Use these if you need more explanations!
- https://www.javascripttutorial.net/es6/
- https://javascript.info/

- Use this if you need more specific answers!
- https://stackoverflow.com/