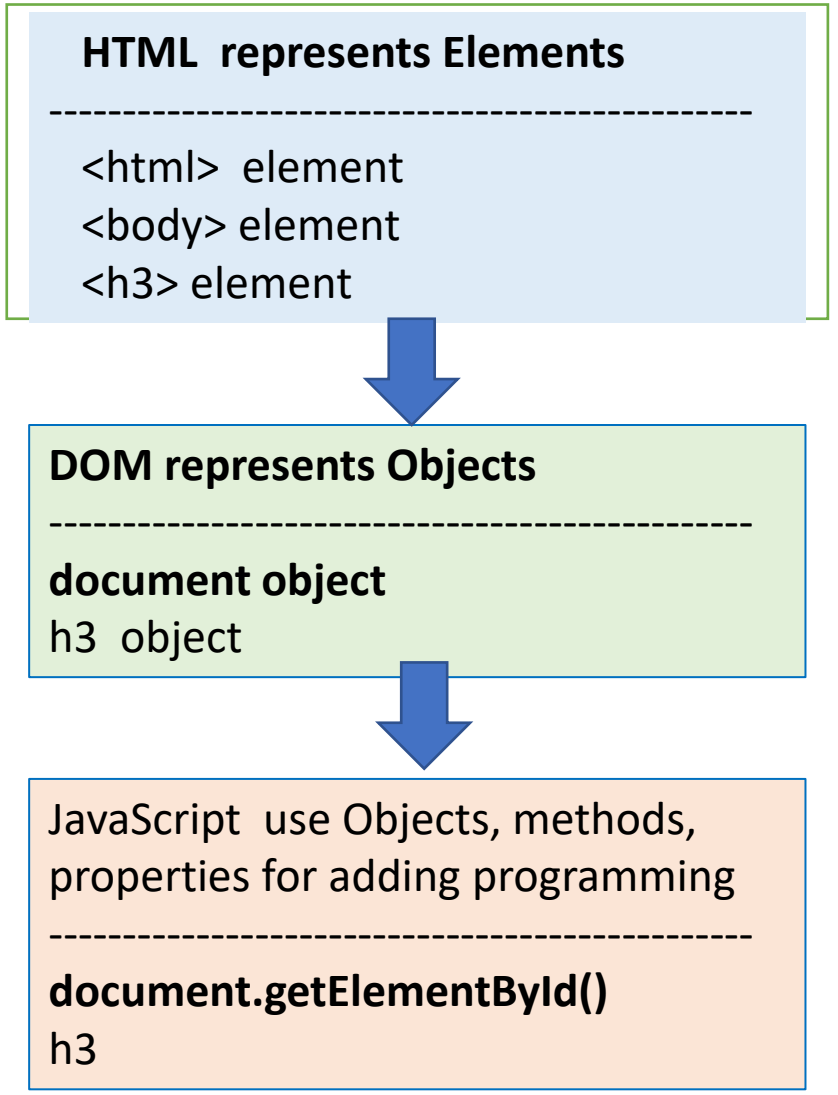
jQuery and DOM

The HTML DOM (Document Object Model)

- It is a **programming interface for HTML documents.**
- It represents the page so that **programs can change the document structure, style, and content.**
- The DOM represents the document as **nodes and objects**; that way, **programming languages can interact with the page.**

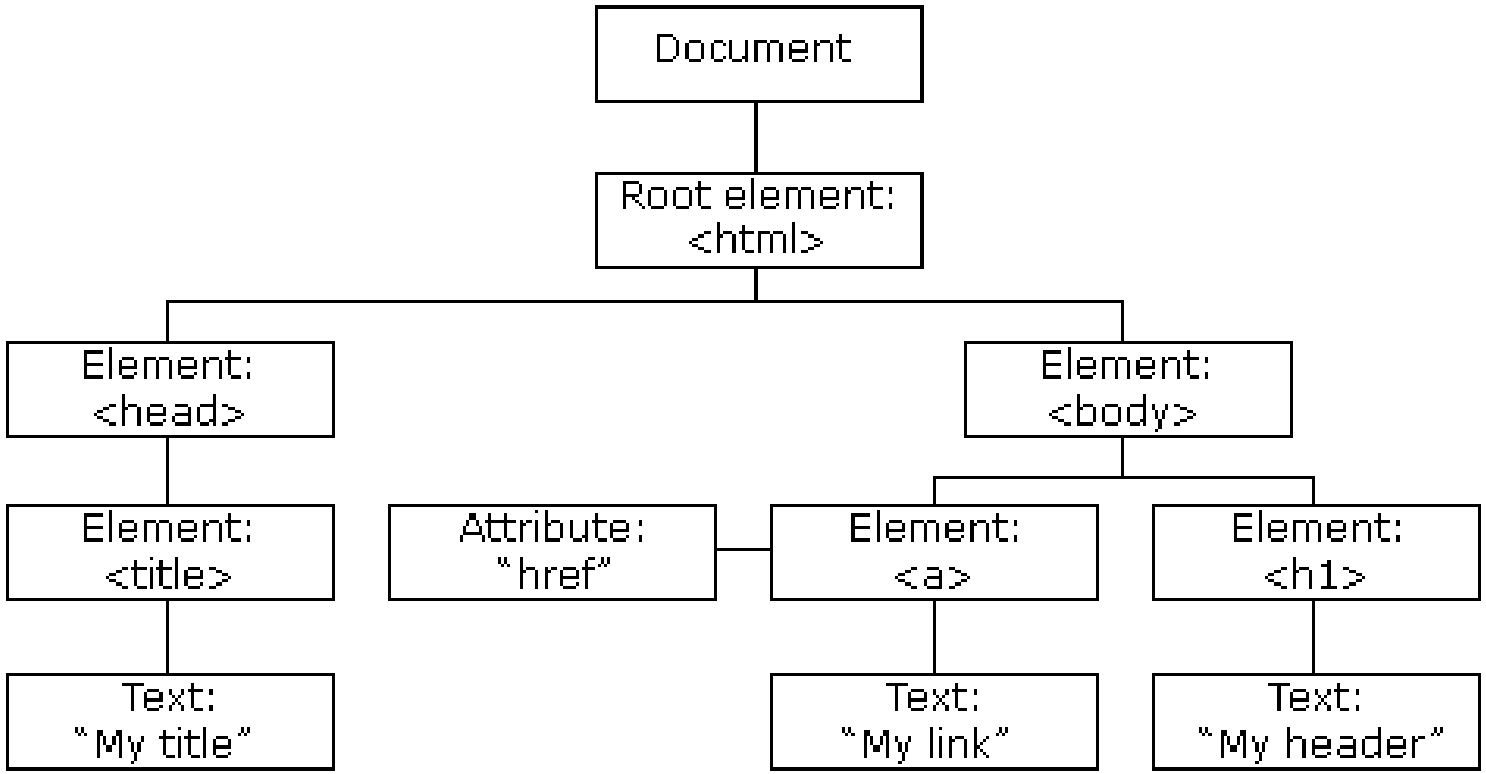


Why DOM is required?

- HTML is used to structure the web pages and Javascript is used to **add behavior** to our web pages.
- When an HTML file is loaded into the browser, the **javascript can not understand the HTML document directly**. So, a corresponding **document is created(DOM)**.
- DOM is basically the **representation of the same HTML document but in a different format with the use of objects.**
- Javascript interprets DOM easily i.e javascript can not understand the tags(<h1>H</h1>) in HTML document but can **understand object h1 in DOM.**

The DOM as a Tree Structure

- **The HTML DOM (Document Object Model)**
- **When a web page is loaded, the browser creates a Document Object Model of the page.**
- **The HTML DOM model is constructed as a tree of Objects:**



The document keyword

- The HTML DOM is a standard **object model** and programming interface for HTML.

It defines:

  - The HTML **elements as objects**
  - The **properties of all HTML elements**
  - The **methods to access all HTML elements**
  - The events for all HTML elements

i.e The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

- **The document object represents your web page.**
- **To access any element in an HTML page, always start with accessing the document object.**
- **Below are some examples of how you can use the document object to access and manipulate HTML.**

1) Finding HTML Elements (Selecgting an HTML Element)

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

The document keyword

2) Changing HTML Elements (Modifying HTML Elements)

Property	Description
element*.innerHTML = *new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element*.style.*property = new style	Change the style of an HTML element
Method	**Description**
element*.setAttribute*(attribute, value)	**Change the attribute value of an HTML element**

3) Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Selecting Elements

1) Finding HTML Element by Id:

getElementById() method is used to find an element by its ID

- The easiest way to find an HTML element in the DOM, is by using the **element id**
- `var el = document.getElementById("intro");`
- If the element is found, the **method will return the element as an object** (**el**).
- If the element is not found, **el** will contain null.

- The below example changes the content (the `innerHTML`) of the `<p>` element with `id="demo"`

```
<!DOCTYPE html>
<html>
<body>
<h2>My First Page</h2>
<p id="demo"></p
<script>
document.getElementById("demo").innerHTML =
"Hello World!";
</script>

</body>
</html>
```

Selecting Elements

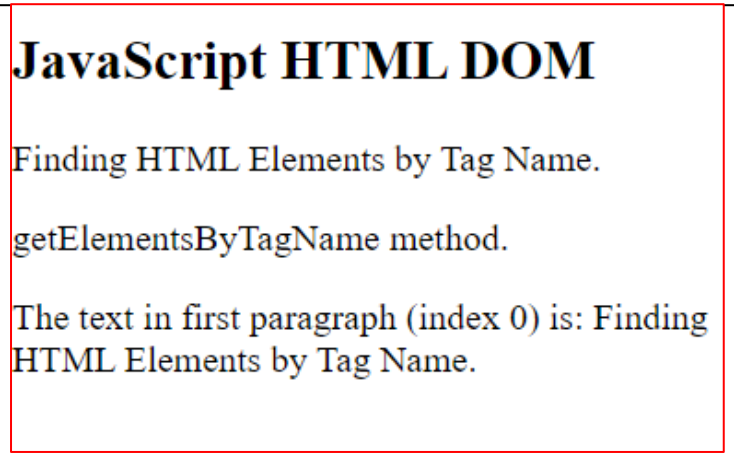
2) Finding HTML Element by Tag Name

```
getElementsByTagName("p");
```

- find an HTML element in the DOM, is by using the **Tag Name**

```
var x = document.getElementsByTagName("p");
```

- If the element is found, the **method will return the element as an object** (in x).
- If the element is not found, **x will contain null.**




```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript HTML DOM</h2>
<p>Finding HTML Elements by Tag Name.</p>
<p>getElementsByTagName method.</p>
<p id="demo"></p>
<script>
const element =
document.getElementsByTagName("p");
document.getElementById("demo").innerHTML = 'The text
in first paragraph (index 0) is: ' +
element[0].innerHTML;

</script>

</body>
</html>
```

DOM- Modify Elements/ changing HTML Content

i) Changing HTML Content using innerHTML

- The easiest way to **modify the content** of an HTML element is by using the **innerHTML** property.
- To change the content of an HTML element, use this syntax:
- **document.getElementById(*id*).innerHTML = *new HTML***

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML="New text!";
</script>

</body>
</html>
```

DOM- Modify Elements/ changing HTML Content

2) Changing the Value of an Attribute

`getElementById(id).attribute`

- To change the value of an HTML attribute, use this syntax:

- `document.getElementById(id).attribute = new value`

```
<!DOCTYPE html>
<html>
<body>

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landsc
ape.jpg";
</script>

</body>
</html>
```

Style an element

Changing HTML Style (Changing CSS)

- The HTML DOM allows JavaScript to change the style of HTML elements.
- To change the style of an HTML element, use this syntax:

```
document.getElementById(id).style.property =
 new style
```

```
<html>
<body>

<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color="blue";
</script>

</body>
</html>
```

Creating and insert elements

createElement() and appendElement():

- The **createElement()** method creates a new node (element).
- The **appendChild()** method appends a node (element) as the last child of an element.

```
<html>
<body>
<h1>Creating and appending Element</h1>
<ul id="myList">
  <li>Coffee</li>
  <li>Tea</li>
</ul>

<p>Click "Append" to append an item to the
end of the list:</p>

<button
onclick="myFunction()">Append</button>
```

```
<script>
function myFunction()
{
// Create an "li" node:
const node = document.createElement("li");

// Create a text node:
const textnode = document.createTextNode("Water");

// Append the text node to the "li" node:
node.appendChild(textnode);

// Append the "li" node to the list:
document.getElementById("myList").appendChild(node);
}
</script>
</body>
</html>
```

Create a New Element Using jQuery

- Like most jQuery operations, **creating an element starts with the dollar function, $( ).**
  $("<a>")
- In this case, there's **a single object representing an "a" element which we just created**.

Adding More Complex HTML:

- The dollar function can create more than one element.
- In fact, it can **build any tree of HTML elements**

```
$("<ul><li>one</li><li>two</li><li>three</li></ul>")

$('<img src="photo.jpg" alt="my hometown" />')
```

Set Attributes on a New Element:

- ```
  photo = new Date().getHours() > 12 ? "afternoon.jpg" : "morning.jpg";
  $("<img>", { src: photo });
  ```
  ```

jQuery append() Method

- The jQuery **append()** method **inserts content** AT THE END of the selected HTML elements.

```
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/
jquery/3.6.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#btn1").click(function(){
    $("p").append(" <b>Appended
text</b>.");
  });

  $("#btn2").click(function(){
    $("ol").append("<li>Appended
item</li>");
  });
});
</script>
</head>
<body>
```

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

<ol>
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ol>

<button id="btn1">Append text</button>
<button id="btn2">Append list
items</button>

</body>
</html>
```

HTML DOM Element remove()

- The remove() method removes an element (or node) from the document.

```
<html>
<body>
<h1>The Element Object</h1>
<h2>The remove() Method</h2>

<p id="demo">Click "Remove", and this paragraph will be
removed from the DOM.</p>
<button onclick="myFunction()">Remove</button>

<script>
function myFunction() {
  const element = document.getElementById("demo");
  element.remove();
}
</script>
</body>
</html>
```

jQuery - Remove an Element

- **The jQuery remove()** method removes the selected element(s) and its child elements.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jque
ry/3.6.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").remove();
  });
});
</script>
</head>
<body>

<div id="div1"
style="height:100px;width:300px;border:1px
solid black;background-color:yellow;">
This is some text in the div.
<p>This is a paragraph in the div.</p>
<p>This is another paragraph in the div.</p>

</div>
<br>

<button>Remove div element</button>

</body>
</html>
```

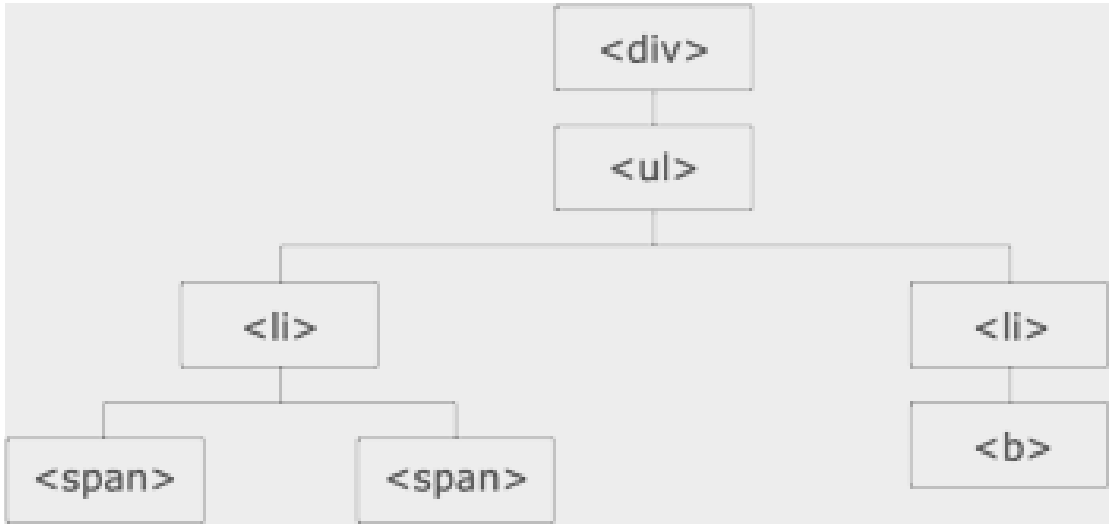
Traversing the DOM

- **jQuery traversing,** which means "**move through**", are used to **"find" (or select) HTML elements based on their relation** to other elements.

- Start with one selection and move through that selection until you reach the elements you desire.

- The image below illustrates an HTML page as a tree (DOM tree).
- With jQuery traversing, we can **easily move up (ancestors), down (descendants) and sideways (siblings) in the tree**, starting from the selected (current) element.

- This **movement is called traversing** - or moving through - the DOM tree.



- The ** ** element is **parent** of **<ul>,** and an **ancestor** of everything inside of it
- The <ul> is the **parent** of both <li> elements, and a **child** of
- T <li> element is the **parent** of , **child** of <ul> and a **descendant** of
- The element is a **child** of the left <li> and a **descendant** of <ul> and
- The two <li> elements are **siblings** (they share the same parent)

Traversing the DOM- jQuery Traversing Methods

Traversing the DOM:

- Query provides a **variety of methods that allow us to traverse the DOM**.
- The largest category of traversal methods are **tree-traversal**

Ancestors:

parent() → gives **parent element of specified selector**

Syntax:

$(selector).parent();

parents()→ it gives all ancestor elements of the specified selector.

Syntax:

$(selector).parents();

)

parentsUntil()→ it gives **all ancestor elements** between specified selector and arguments.

Syntax:

- $(selector).parentsUntil(selector, filter element)
- $(selector).parentsUntil(element, filter element

offsetParent()→it gives the first positioned parent element of specified selector.

Syntax:

- $(selector).offsetParent();

closest()→it gives the first ancestor of the specified selector.

Syntax:

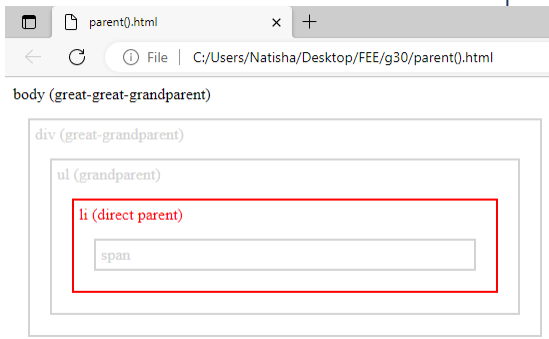
- $(selector).closest(selector);
- $(selector).closest(selector, context);
- $(selector).closest(selection);
- $(selector).closest(element);

Traversing the DOM- jQuery Traversing Methods

jQuery parent() Method:

- The **parent() method** returns the direct parent element of the selected element.
- The DOM tree: This method only traverse a **single level up the DOM tree**.
- To traverse all the way up to the document's root element (to return grandparents or other ancestors), use the parents() or the parentsUntil() method.

```
<html><head>
<style>
.ancestors * {
  display: block;
  border: 2px solid lightgrey;
  color: lightgrey;
  padding: 5px;
  margin: 15px;
}
</style>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("span").parent().css({"color": "red", "border": "2px solid red"});
});
</script>
</head>
<body class="ancestors">body (great-great-grandparent)
  <div style="width:500px;">div (great-grandparent)
    <ul>ul (grandparent)
      <li>li (direct parent)
        <span>span</span>
      </li>
    </ul>
  </div>
</body></html>
```



Traversing the DOM- jQuery Traversing Methods

Descendants:

children() →it gives the children of each selected elements, optionally filtered by a selector.

Syntax:

$(selector).children();

find()→

- it gives descendant elements of specified elements, filtered by a selector, jQuery object, or element.
- Syntax:

$(selector).find('selector to find');

Siblings:

prevAll() → it gives all previous sibling elements of the specified selector.

Syntax:

$(selector).prevAll(selector, filter element)

$(selector).prevAll(element, filter element)

siblings()→ it gives all siblings of the specified selector.

Syntax:

- $(selector).siblings();

- **next()→** it gives the next sibling element of the specified selector.
  Syntax:$(selector).next();

- **nextAll()→**it gives all next sibling elements of the specified selector.
  Syntax:$(selector).nextAll();

- **nextUntil()→**it gives all next sibling elements between specified selector and arguments.
  Syntax:$(selector).nextUntil();

- **prev()→**it gives the previous sibling element of the specified selector.
  Syntax:$(selector).prev(selector); $(selector).prev()

jQuery children() Method

jQuery children() Method:

- The children() method **returns all direct children of the selected element.**

```
<html>
<head>
  <title>jQuery children() Method Example</title>
  <style>
    .highlight {
      background-color: yellow;
    }
  </style>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script>
    $(document).ready(function() {
      // Select the parent element
      var parentElement = $('#parent');
```

```
// Use the children() method to select all direct child paragraphs
      var childElements = parentElement.children('p');

      // Add a CSS class to the child elements
      childElements.addClass('highlight');
    });
  </script>
</head>
<body>
  <div id="parent">
    <h2>Parent Element</h2>
    <p>First Child</p>
    <p>Second Child</p>
    <p>Third Child</p>
  </div>
</body>
</html>
```

jQuery siblings() Method

jQuery siblings() Method:

- The siblings() method **returns all siblings() of the selected element.**

```
<html>
<head>
  <title>jQuery siblings() Method
Example</title>
  <style>
    .highlight {
      color: red;
    }
  </style>
  <script
src="https://code.jquery.com/jquery-
3.6.0.min.js"></script>
  <script>
    $(document).ready(function() {
      // Select an element
      var element = $('#target');
      // Use the siblings() method to select all
sibling elements
      var siblingElements = element.siblings();

      // Add a CSS class to the sibling
elements
      siblingElements.addClass('highlight');
    });
  </script>
</head>
<body>
  <h2>Parent Element</h2>
  <p>Sibling 1</p>
  <p id="target">Target Element</p>
  <p>Sibling 2</p>
</body>
</html>
```

Traversing the DOM- jQuery Traversing Methods

Filtering

first() → it gives the first element of the specified selector.

- Syntax:
- $(selector).first();

last() → it gives the last element of the specified selector.

- Syntax:
- $(selector).last();

eq()→it gives an element with a specific index number of the specified selector.

- Syntax:
- $(selector).eq(index);
- $(selector).eq(indexFromEnd);

filter()→it remove/detect an elements that are matched with specified selector.

- Syntax:
- $(selector).filter(selector)
- $(selector).filter(function)
- $(selector).filter(selection)
- $(selector).filter(elements)

has()→ it gives all elements that have one or more elements within, that are matched with specified selector.

- Syntax:
- $(selector).has(selector);

is()→ it checks if one of the specified selector is matched with arguments.

- Syntax:
- .is(selector)
- .is(function)
- .is(selection)
- .is(elements)

map() →Pass each element in the current matched set through a function, producing a new jQuery object containing the return values

- Syntax:

.map(callback)

slice() →it selects a subset of specified selector based on its argument index or by start and stop value.

- Syntax:
- $(selector).slice(start, end);
- $(selector).slice(start);

jQuery filter() Method

jQuery filter() Method:

- The filter() method in jQuery is used to narrow down the set of matched elements based on a specific criteria

```
<!DOCTYPE html>
<html>
<head>
  <title>jQuery filter() Method Example</title>
  <style>
    .highlight {
      background-color: yellow;
    }
  </style>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script>
    $(document).ready(function() {
      // Select all paragraphs
      var paragraphs = $('p');
      // Use the filter() method to select paragraphs with a certain class
      var filteredElements = paragraphs.filter('.special');

      // Add a CSS class to the filtered elements
      filteredElements.addClass('highlight');
    });
  </script>
</head>
<body>
  <h2>Paragraphs Example</h2>
  <p>First Paragraph</p>
  <p class="special">Second Paragraph</p>
  <p>Third Paragraph</p>
  <p class="special">Fourth Paragraph</p>
</body>
</html>
```

jQuery first() Method

jQuery first() Method:

- it gives the first element of the specified selector.
- first() method on $('p') to select the first p element and store it in the firstParagraph variable.

```
<html>
<head>
  <title>jQuery first() Method
Example</title>
  <style>
    .highlight {
      background-color: yellow;
    }
  </style>
  <script
src="https://code.jquery.com/jquery-
3.6.0.min.js"></script>
  <script>
    $(document).ready(function() {
      // Select the first paragraph
      var firstParagraph = $('p').first()
```

```
// Add a CSS class to the first paragraph
      firstParagraph.addClass('highlight');
    });
  </script>
</head>
<body>
  <h2>Paragraphs Example</h2>
  <p>First Paragraph</p>
  <p>Second Paragraph</p>
  <p>Third Paragraph</p>
</body>
</html>
```

jQuery first() Method

jQuery last() Method:

- it gives the last element of the specified selector.
- first() method on $('p') to select the first p element and store it in the last Paragraph variable.

```
<html>
<head>
  <title>jQuery last() Method
Example</title>
  <style>
    .highlight {
      background-color: yellow;
    }
  </style>
  <script
src="https://code.jquery.com/jquery-
3.6.0.min.js"></script>
  <script>
    $(document).ready(function() {
      // Select the last paragraph
      var lastParagraph = $('p').last();
```

```
// Add a CSS class to the last paragraph
      lastParagraph.addClass('highlight');
    });
  </script>
</head>
<body>
  <h2>Paragraphs Example</h2>
  <p>First Paragraph</p>
  <p>Second Paragraph</p>
  <p>Third Paragraph</p>
</body>
</html>
```