
PHOENIX
JS

General

PhoenixJS is a modern JavaScript helper library. It is a library which encapsulates manipulation of the DOM. If you have any experience working with *jQuery* this should sound familiar.

Functionality

PhoenixJS makes use of **_p**, e.g. **_p('.my-container').lastSibling();**. The **_p** function should accept any valid CSS query selector and return a **PhoenixElement**. This **PhoenixElement** holds all the **nodes** which resulted from the CSS query. All functions should return a **PhoenixElement** so the returned value can be chained for further applying of functions. If no elements result from the CSS query selector a **PhoenixElement** should still be returned, but all functions applied to that **PhoenixElement** will have no effect because the element list is empty. If there are more than one elements in the list the functions should apply to all the elements in the list, if not otherwise stated. These are the functions that should be implemented:

1. **(12.5%)** Implement a function called l**astSibling** which should return the last sibling of the current node. If the current node is the last sibling it should return itself.

```
<div class="my-container">
    <p id="1"></p>
    <p id="2"></p>
    <p id="3"></p>
    <p id="4"></p>
    <p id="5"></p>
</div>

// PhoenixElement containing <p id="5"></p>
_p('.my-container #1').lastSibling();
```

2. **(12.5%)** Implement a function called **on** which should accept two parameters: *eventType* and *eventHandler*. This should be a generic input handler register function.

```
<form action="" id="my-form">
    <input type="text" id="username" />
    <input type="password" id="password" />
    <input type="submit" value="Submit" />
</form>

_p('#my-form').on('submit', function (e) {
    e.preventDefault();
    // Do something..
});
```

3. **(12.5%)** Implement a function called **firstChild** which should return the first child of the current node. If the current node contains no children it should return itself.

```
<div className="container">
    <p id="pg-1"></p>
    <p id="pg-2"></p>
    <p id="pg-3"></p>
</div>

// A PhoenixElement containing <p id="pg-3"></p>
_p('.container').firstChild();
```

4. **(12.5%)** Implement a function called **pushElements** which accepts 1 to N arguments which should be a **string**. The arguments should be added to the end of the element.

```
// Before
<div class="my-container"></div>

_p('.my-container').pushElements('<h1>Title<h1>', '<p>My paragraph</p>');

// After
<div class="my-container">
    <h1>Title</h1>
    <p>My paragraph</p>
</div>
```

5. **(12.5%)** Implement a function called **wrapAround** which accepts either a **string** or **Node** which should wrap around the selected node.

```
// Before
<div class="inner"></div>

_p('.inner').wrapAround('<div class="outer"></div>');

// OR

var outer = document.createElement('div');
outer.className = 'outer';
_p('.inner').wrapAround(outer);

// After
<div class="outer">
    <div class="inner"></div>
</div>
```

6. **(12.5%)** Implement a function called **insertAtIndex** which accepts two arguments: **index** (a number indicating the index, this is zero-based) and **string / Node**. It should insert the node within the parent in a particular index. If the index is out of bounds it should throw an exception, that is done with the **throw** keyword.

```
<div class="container">
    <p id="p-1"></p>
    <p id="p-2"></p>
    <p id="p-3"></p>
</div>

_p('.container').insertAtIndex(0, '<p id="p-0.5"></p>');

<div class="container">
    <p id="p-0.5"></p>
    <p id="p-1"></p>
    <p id="p-2"></p>
    <p id="p-3"></p>
</div>
```

7. **(12.5%)** Implement a function called **trigger** which accepts one parameter: **eventType (string)** and should trigger the event for the current selected node.

```
<button id="my-btn">Click</button>

// The button should be clicked
_p('#my-btn').trigger('click');
```

8. **(12.5%)** Implement a function called **containsDataset** which accepts one parameter: **datasetName (string)** and determines if the element contains a specific **dataset** as attribute.

```
<div class="my-div" data-info="secret"></div>

_p('.my-div').containsDataset('info'); // true
_p('.my-div').containsDataset('data-info'); // false
_p('.my-div').containsDataset('other'); // false
```

Other

All implementations should use **JavaScript** and no external libraries are allowed