Write JavaScript for the Web

10 hours Medium

Last updated on 4/2/20

Manipulate the DOM

Nice! You passed this exercise!

Evaluated skills

- capture DOM events
- access and modify DOM elements

Question 1

Which of the following functions will return a collection containing all `<p>` nodes within the document?

- ✓ (●) Both `document.getElementsByTagName('p');` and `document.querySelectorAll('p');`
- ( ) Both `document.getElementsByClassName('p');` and `document.getElementById('p');`
- ( ) Only `document.querySelector('p');`

`document.getElementsByTagName('p')` and `document.querySelectorAll('p')` both return the required collection. `document.getElementsByClassName('p')` returns a collection of all nodes with `class="p"`, `document.getElementById('p')` returns the node with `id="p"`, and `document.querySelector('p')` will only return the first `<p>` node in the document.

Question 2

Select the three functions below which will return the paragraph `<p>` with id `main-article`.

Careful, there are several correct answers.

- [ ] `document.getElementsByTagName('main-article');`
- ✓ [x] `document.getElementById('main-article');`
- ✓ [x] `document.querySelector('p#main-article');`
- [ ] `document.querySelector('p.main-article');`
- ✓ [x] `document.querySelector('#main-article');`

As there can only be one node with any given `id`, both `document.getElementById('main-article')` and `document.querySelector('#main-article')` will return the required paragraph, as will `document.querySelector('p#main-article')`

. `document.getElementsByTagName('main-article')` *will return a collection of* `<main-article>` *nodes, and* `document.querySelector('p.main-article')` *will return the first node of type* `<p class="main-article">` .

Question 3

Which of the following functions will return a collection containing all `<li>` nodes (and ONLY `<li>` nodes) of class `list-item-active` ?

- ○ `document.getElementById('list-item-active');`
- ✓ ◉ `document.querySelectorAll('li.list-item-active');`
- ○ `document.getElementsByClassName('list-item-active');`
- ○ `document.querySelector('li.list-item-active');`

`document.querySelectorAll('li.list-item-active')` *will return the required collection.* `document.getElementById('list-item-active')` *will return the node with* `id="list-item-active"` , `document.getElementsByClassName('list-item-active")` *will return all nodes with* `class="list-item-active"` *, regardless of tag name, and* `document.querySelector('li.list-item-active')` *will only return the first node of type* `<li class="list-item-active">` .

Question 4

Which of the following would add the class `"active"` to the paragraph with id `"new-article"` ?

- ○ `document.getElementsByClassName('new-article').classList.push('active');`
- ○ `document.getElementById('new-article').classList.push('active');`

- ✓ ● `document.querySelector('#new-article').classList.add('active');`
- ○ `document.querySelector('new-article').classList.add('active');`

Although `document.getElementById('new-article')` *accesses the correct paragraph, the correct method for adding a class to a* `classList` *is* `add()` *and not* `push()` *. The other answers do not access the correct paragraph.*

Question 5

Which of the following would replace the current text of the paragraph with id `article` **with the string** `'new content'` **?**

- ○ `document.getElementById('article').textContent += 'new content';`
- ✓ ● `document.getElementById('article').textContent = 'new content';`
- ○ `document.getElementById('article') += 'new content';`
- ○ `document.getElementById('article').textContent('new content');`

`document.getElementById('article').textContent += 'new content'` *would append the string "new content" to the current content of the paragraph rather than replace it, and the other two methods are simply incorrect. The correct answer is* `document.getElementById('article').textContent = 'new content'`

Question 6

Which of the following would remove the button with id `submit-form` **?**

- ○

javascript

```
1 document.querySelector('#submit-form').delete();
```

javascript
```
const button = document.querySelector('#submit-form');
button.removeChild('button');
```

javascript
```
const button = document.querySelector('#submit-form');
button.parentNode.removeChild(button);
```

javascript
```
const button = document.querySelector('#submit-form');
button.delete(button);
```

Remember that, to remove a node, you need access both to the node itself and to its parent; you cannot simply remove a node with its reference.

Question 7

Which of the following adds a new list item containing the string `Apples` to the unordered list with id `shopping-list` ?

javascript
```
let newItem = document.createElement('li');
newItem.textContent = 'Apples';
document.querySelector('ul#shopping-list').appendChild(newItem);
```

javascript
```
let newItem = document.createElement('<li>Apples</li>');
document.querySelector('ul#shopping-list').appendChild(newItem);
```

javascript

```
let newItem = document.createNode('li');
newItem.textContent = 'Apples';
document.querySelector('ul#shopping-list').appendChild(newItem);
```

The three steps to adding a new child node to an element are:

1. *Create the element*
2. *Populate the element*
3. *Add the finished element to the DOM*

Here, answer B does not use the `createElement()` *function correctly, and answer C uses the incorrect* `createNode()` *function*

Question 8

Which of the following will make the button with id `submit-form` **, when clicked, display an alert?**

javascript

```
document.getElementById('submit-form').listen('click', () => {
    alert('Form submitted!');
});
```

javascript

```
document.getElementById('submit-form').addEventListener('click', () => {
    alert('Form submitted!');
});
```

javascript

```
document.getElementById('submit-form').onclick(function() {
    alert('Form submitted!');
});
```

javascript

```
document.addEventListener('#submit-form', 'click', () => {
    alert('Form submitted!');
});
```

To add an event listener to an element:

1. Access the element

2. Use the addEventListener() method, passing it the event type to listen for and the function to call when the event takes place

Answers A and C use the incorrect functions `listen()` *and* `onclick()` *, and answer D attempts to add an event listener incorrectly.*

Question 9

Which of the following will log the cursor's position (its X and Y coordinates) to the console as it moves over the `<div>` **element of id** `my-map` **?**

javascript

```
const mapDiv = document.getElementById('my-map');
mapDiv.addEventListener('mouseover', ($event) => {
    console.log('X: ' + $event.offsetX + ' | Y: ' + $event.offsetY);
});
```

javascript

```
const mapDiv = document.getElementById('my-map');
mapDiv.addEventListener('mouseover', ($event) => {
    console.log('X: ' + offsetX + ' | Y: ' + offsetY);
});
```

javascript

```
const mapDiv = document.getElementById('my-map');
mapDiv.addEventListener('mousemove', ($event) => {
```

```
    console.log('X: ' + $event.offsetX + ' | Y: ' + $event.offsetY);
});
```

Answer A will only register the mouse's position once, as it arrives into the selected div (the `mouseover` event). Answer B uses the incorrect event and does not correctly output event information. Answer C uses the correct event, but does not output the information correctly, attempting to access an inexistent `mouse` property.

Question 10

Challenge time! Which of the following will, whenever the user scrolls the window, log the distance between the top of the window and the top of the `<div>` element with id `results` ?

Hint: you will need to do a bit of research for this one!

○

javascript
```
const resultsDiv = document.querySelector('div#results');
window.addEventListener('scroll', () => {
    console.log(resultsDiv.scrollTop - window.scrollTop);
});
```

✓ ○

javascript
```
const resultsDiv = document.querySelector('div#results');
window.addEventListener('scroll', () => {
    console.log(resultsDiv.offsetTop - window.scrollY);
});
```

✗ ◉

javascript
```
const resultsDiv = document.querySelector('div#results');
window.addEventListener('scroll', () => {
    console.log(window.scrollTop - resultsDiv.scrollTop);
```

```
4 });
```

You need to calculate the difference between `resultsDiv.offsetTop` *(how far the* `<div>` *is from the top of the* ***document****), and* `window.scrollY` *(how far the* ***window****'s content has been scrolled). This was a tricky one!*

N.B. This sort of calculation is extremely useful for things like parallax scrolling effects, as you need to know where an element is with respect to the surrounding window.

SUMMARY

HANDLE INPUT FROM YOUR USERS

Teacher

Will Alexander

Scottish developer, teacher and musician based in Paris.

LEARN MORE

English

Download on the App Store