Write JavaScript for the Web

10 hours Medium

Last updated on 4/2/20

Get data from users and from servers

Nice! You passed this exercise!

Evaluated skills

handle user input

Question 1

Which of the following gives access to the data entered by the user in the input with id `first-name` ?

- ○ `let input = document.getElementById('first-name').textContent;`
- ○ `let input = document.getElementById('first-name').input;`
- ✓ ◉ `let input = document.getElementById('first-name').value;`
- ○ `let input = document.getElementById('first-name').innerHTML;`

The `value` of an input element contains the data entered by the user.

Question 2

Which of the following event listeners will be triggered when the user clicks or tabs into the input with id `email` ?

- ○

```javascript
document.getElementById('email').addEventListener('click', () => {
    console.log('Input activated!');
});
```

- ✓ ◉

```javascript
document.getElementById('email').addEventListener('focus', () => {
    console.log('Input activated!');
});
```

- ○

```javascript
document.getElementById('email').addEventListener('blur', () => {
    console.log('Input activated!');
});
```

- ○

```javascript
```

```
document.getElementById('email').addEventListener('enter', () => {
    console.log('Input activated!');
});
```

The `focus` event is triggered when the user activates the input, no matter how they activate it. The `click` event is only triggered when the user clicks on the input (and is triggered for every click). The `blur` event is triggered when focus shifts away from the input, and there is no `enter` event.

Question 3

Which of the following listeners will log the state of the checkbox with id `available` to the console whenever the user checks or unchecks it?

✓ ◉

javascript

```
document.getElementById('available').addEventListener('change', ($event) => {
    console.log('Available: ' + $event.target.checked);
})
```

○

javascript

```
document.getElementById('available').addEventListener('change', ($event) => {
    console.log('Available: ' + $event.checked);
})
```

○

javascript

```
document.getElementById('available').addEventListener('change', ($event) => {
    console.log('Available: ' + $event.value);
})
```

javascript

```
document.getElementById('available').addEventListener('change', ($event) => {
    console.log('Available: ' + $event.target.value);
})
```

For a checkbox, its checked-or-unchecked state (a boolean) is stored in `$event.target.checked` *, or the* `checked` *key of the* `target` *object within the* `change` *event object.*

Question 4

Which of the following will log the selected value of the group of radio buttons (or dropdown menu) with id `status` **to the console whenever the user chooses an option?**

javascript
```
document.getElementById('status').addEventListener('change', ($event) => {
   console.log('Status: ' + $event.target.checked);
});
```

javascript
```
document.getElementById('status').addEventListener('change', ($event) => {
   console.log('Status: ' + $event.checked);
});
```

✓ javascript
```
document.getElementById('status').addEventListener('change', ($event) => {
   console.log('Status: ' + $event.target.value);
});
```

javascript
```
document.getElementById('status').addEventListener('change', ($event) => {
   console.log('Status: ' + $event.value);
```

```
});
```

For groups of radio buttons or dropdown menus, the `value` *key in the* `change` *event's* `target` *object contains the value of the selected option.*

Question 5

Which of the following checks the strength of a password while it is being typed into the input with id `password-input` **, and prints the corresponding message to the console?**

(Here, we assume that `checkPassword(password)` **returns true if the password is strong, and false if it is weak.)**

- ○

javascript

```javascript
document.getElementById('password-input').addEventListener('focus', ($event) => {
    if (checkPassword($event.target.value)) {
        console.log('Password is weak, make it stronger!');
    }
    else {
        passwordIsInvalid('That\'s more like it! Password is strong!');
    }
});
```

- ○

javascript

```javascript
document.getElementById('password-input').addEventListener('blur', ($event) => {
    if (checkPassword($event.target.value)) {
        console.log('Password is weak, make it stronger!');
    }
    else {
        passwordIsInvalid('That\'s more like it! Password is strong!');
    }
});
```

javascript

```javascript
document.getElementById('password-input').addEventListener('change', ($event) => {
    if (checkPassword($event.target.value)) {
        console.log('Password is weak, make it stronger!');
    }
    else {
        passwordIsInvalid('That\'s more like it! Password is strong!');
    }
});
```

javascript

```javascript
document.getElementById('password-input').addEventListener('input', ($event) => {
    if (checkPassword($event.target.value)) {
        console.log('Password is weak, make it stronger!');
    }
    else {
        passwordIsInvalid('That\'s more like it! Password is strong!');
    }
});
```

The `input` event is triggered every time the user changes the text in the input field. The other answers listen for the wrong events.

Question 6

Which of the following will check the validity of the email address entered in the input with id `email` once the user has clicked or tabbed away from that input, and print the corresponding message to the console?

(Here, we assume that `checkEmail(email)` returns true if the email address is valid, and false if it is not.)

javascript

```javascript
document.getElementById('email').addEventListener('focus', ($event) => {
    if (checkEmail($event.target.value)) {
        console.log('Email address is valid!');
    }
```

```
    else {
        console.log('Invalid email address!')
    }
});
```

javascript

```
document.getElementById('email').addEventListener('blur', ($event) => {
    if (checkEmail($event.target.value)) {
        console.log('Email address is valid!');
    }
    else {
        console.log('Invalid email address!')
    }
});
```

javascript

```
document.getElementById('email').addEventListener('change', ($event) => {
    if (checkEmail($event.target.value)) {
        console.log('Email address is valid!');
    }
    else {
        console.log('Invalid email address!')
    }
});
```

javascript

```
document.getElementById('email').addEventListener('input', ($event) => {
    if (checkEmail($event.target.value)) {
        console.log('Email address is valid!');
    }
    else {
        console.log('Invalid email address!')
    }
});
```

The `blur` event is triggered when the user clicks or tabs away from the input field. The other answers listen for the wrong events.

Question 7

Which of the following will check the validity of this input:

`<input type="text" id="first-name" required>`

and only submit the data if the input is valid, when the form's `submit` button is clicked?

(Here we assume that the element with id `form` is a `<form>`.)

✓ ◉

javascript

```javascript
const firstNameInput = document.getElementById('first-name');
document.getElementById('form').addListener('submit', () => {
    if (firstNameInput.checkValidity()) {
        console.log('Form can be submitted!');
    }
});
```

○

javascript

```javascript
const firstNameInput = document.getElementById('first-name');
document.getElementById('form').addListener('submit', () => {
    if (firstNameInput.isValid()) {
        console.log('Form can be submitted!');
    }
});
```

○

javascript

```javascript
const firstNameInput = document.getElementById('first-name');
document.getElementById('form').addListener('blur', () => {
    if (firstNameInput.validity) {
        console.log('Form can be submitted!');
```

```
    }
});
```

javascript

```
const firstNameInput = document.getElementById('first-name');
document.getElementById('form').addListener('change', () => {
    if (firstNameInput.checkIfValid()) {
        console.log('Form can be submitted!');
    }
});
```

An input element's `checkValidity()` *method checks if its value passes its validation constraints (* `required` , `min` , `max` *, etc.). The methods used in the other answers do not exist.*

Question 8

Which of the following correctly builds a GET request to the OpenWeatherMap API?

javascript

```
let weatherRequest = new XMLHttpRequest('GET', 'http://api.openweathermap.org/data/2.5/weather?q=Paris');
```

✓ javascript

```
let weatherRequest = new XMLHttpRequest();
weatherRequest.open('GET', 'http://api.openweathermap.org/data/2.5/weather?q=Paris');
```

javascript

```
let weatherRequest = new XMLHttpRequest();
weatherRequest.onreadystatechange('GET', 'http://api.openweathermap.org/data/2.5/weather?q=Paris');
```

javascript

```
let weatherRequest = new XMLHttpRequest();
weatherRequest.send('GET', 'http://api.openweathermap.org/data/2.5/weather?q=Paris');
```

To build and prepare an XMLHttpRequest, you must first create a new XMLHttpRequest object, and then use its `open` *method to set its verb and URL.*

Question 9

Which of the following will correctly request a blog post from a (fictitious) server and then display that post in the DOM?

(Here we imagine that the server will send back a JSON object containing `title` **and** `content` **keys)**

javascript

```
const postsSection = document.getElementById('posts-section');

let postRequest = new XMLHttpRequest();
postRequest.open('GET', 'http://my-blog-api.com/post?id=4acc432mij');

postRequest.onreadystatechange = () => {
   if (postRequest.readyState === 4) {
      let newArticle = document.createElement('article');
      let newTitle = document.createElement('h2');
      let newContent = document.createElement('p');

      const response = postRequest.response;

      newTitle.textContent = response['title'];
      newContent.textContent = response['content'];

      newArticle.appendChild(newTitle);
      newArticle.appendChild(newContent);

      postsSection.appendChild(newArticle);
```

```
    }
};

postRequest.send();
```

javascript

```
const postsSection = document.getElementById('posts-section');

let postRequest = new XMLHttpRequest();
postRequest.open('GET', 'http://my-blog-api.com/post?id=4acc432mij');

postRequest.onreadystatechange(() => {
    if (postRequest.readyState === 4) {
        let newArticle = document.createElement('article');
        let newTitle = document.createElement('h2');
        let newContent = document.createElement('p');

        const response = JSON.parse(postRequest.response);

        newTitle.textContent = response['title'];
        newContent.textContent = response['content'];

        newArticle.appendChild(newTitle);
        newArticle.appendChild(newContent);

        postsSection.appendChild(newArticle);
    }
});

postRequest.send();
```

javascript

```
const postsSection = document.getElementById('posts-section');

let postRequest = new XMLHttpRequest();
```

```
postRequest.open('GET', 'http://my-blog-api.com/post?id=4acc432mij');

postRequest.onreadystatechange = () => {
    if (postRequest.readyState === 4) {
        let newArticle = document.createElement('article');
        let newTitle = document.createElement('h2');
        let newContent = document.createElement('p');

        const response = JSON.parse(postRequest.response);

        newTitle.textContent = response['title'];
        newContent.textContent = response['content'];

        newArticle.appendChild(newTitle);
        newArticle.appendChild(newContent);

        postsSection.appendChild(newArticle);
    }
};

postRequest.send();
```

○

javascript

```
const postsSection = document.getElementById('posts-section');

let postRequest = new XMLHttpRequest();
postRequest.open('GET', 'http://my-blog-api.com/post?id=4acc432mij');

postRequest.onreadystatechange = () => {
    let newArticle = document.createElement('article');
    let newTitle = document.createElement('h2');
    let newContent = document.createElement('p');

    const response = JSON.parse(postRequest.response);

    newTitle.textContent = response['title'];
    newContent.textContent = response['content'];

    newArticle.appendChild(newTitle);
```

```
    newArticle.appendChild(newContent);

    postsSection.appendChild(newArticle);
};

postRequest.send();
```

To correctly request and handle JSON from a back-end:

1. *Create a* `new` *request object.*
2. *Use its* `open` *method to give it a verb and a URL.*
3. *Implement its* `onreadystatechange` *function, telling it what to do with the response.*
4. *Send the request.*

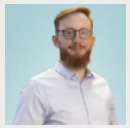
Within the `onreadystatechange` *function:*

1. *Check that the response is ready (* `readyState === 4` *).*
2. *Parse the response with JSON.parse() to create a JavaScript object.*
3. *Use the object's contents to build and append your element(s) to the DOM.*

SUMMARY

UNDERSTAND ASYNCHRONOUS PROGRAMMING

Teacher



Will Alexander

Scottish developer, teacher and musician based in Paris.

OPENCLASSROOMS

BUSINESS SOLUTIONS

CONTACT

LEARN MORE

English