Form Validation and Sanitization with Javascript and Python

Eli the Computer Guy - November 21, 2024
Version 1

Sanitization and Validation

Sanitization is the process of removing harmful text from variable values.

Validation is the process of confirming that dat i formatted the way it needs to be.

In a full web application sanitization and validation occur across the stack. Using javascript on the front end, using Python on the back end, and using strict data types in the database.

HTML and Javascript Element Validation - validity.valid

You can check the validity of an HTML 5 form entry based on type.
- email
- number
- url
- text

VARIABLE.validity.valid - checks if the variable fits with the input type designated on HTML Form.

input-validation.html
```
<input type="email" id="input" required>
<button onclick="validate()">Validate</button>

<script>
    function validate() {
        const input = document.getElementById("input");
        if (input.validity.valid) {
            alert("Valid -- " + input.value);
        } else {
            alert("NOT Valid -- " + input.value);
        }
    }
</script>
```

form-validation.html

```
<form>
    Name: <input type="text" id="name" required><br><br>
    Age: <input type="number" id="age" required><br><br>
    Email: <input type="email" id="input" required><br><br>
    URL: <input type="url" id="website" required><br><br>
    <input type="submit" value="Submit" onclick="validate(event)">
</form>

<script>
    function validate(event) {
        event.preventDefault();

        const name = document.getElementById("name");
        const age = document.getElementById("age");
        const email = document.getElementById("input");
        const website = document.getElementById("website");

        let message = "";

        if (!name.validity.valid) {
            message += "Name is not valid\n";
        }

        if (!age.validity.valid) {
            message += "Age is not valid\\n";
        }

        if (!email.validity.valid) {
            message += "Email is not valid\n";
        }

        if (!website.validity.valid) {
            message += "URL is not valid\n";
        }

        const messageDiv = document.getElementById("message");
        if (message) {
            alert(message);
        } else {
            alert("Form Valid");
            //event.target.submit();
        }
    }
</script>
```

Javascript - Sanitize with RegEx

RegEx (Regular Expressions) allow you to delete or replace unwanted characters with pattern matching.

```
<input type="text" id="input" required>
<button onclick="sanitize()">Validate</button>

<script>
    function removeHTMLTags(input) {
            return input.replace(/<\/?[^>]+(>|$)/g, "");
        }

    function sanitize(){
        const input = document.getElementById("input");
        const clean = removeHTMLTags(input.value);

        alert("Original -- " + input.value + "\nClean -- " + clean);
    }
</script>
```

Python - Validate Input with Validators Module

```
python3 -m pip install validators
```

This script takes a value from the user and states whether it is a URL, Email Address, Domain Name, or IPv4 Address.

valid.py

```
import validators

def is_valid_input(input):
    if validators.url(input):
        print(f'{input} is a URL')
    elif validators.email(input):
        print(f'{input} is an Email')
    elif validators.domain(input):
        print(f'{input} is a Domain')
    elif validators.ipv4(input):
        print(f'{input} is an IP')
    else:
        print(f'{input} is not VALID')

while True:
    value = input("Input: ")
    is_valid_input(value)
```

Python - Sanitize Input with replace()

You can use the replace() method to remove specific characters. This is good to prevent injection attacks.

sanitize-replace.py

```
import os

command = 'ping -c 1 '

while True:
    site = input('Domain Name: ')
    #site = site.replace(';','')

    response = os.popen(f'{command} {site}').read()

    print(response)
```

Python - Escape HTML Tags

You can escape HTML tags so that they do not function, but are still retained with sanitize_html() from the html module.

html-escape.py

```
import html

def sanitize_html(input_string):
    return html.escape(input_string)

while True:
    value = input('User Input: ')
    print(sanitize_html(value))
```

Python - Sanitize HTML with Bleach Module

```
python3 -m pip install bleach
```

The leach python module can be used to remove HTML and Javascript from variable values.

bleach-sanitize.py

```
import bleach

def sanitize_with_bleach(input_string):
    return bleach.clean(input_string, tags=[], attributes={}, strip=True)

unsafe_string = '<a href="http://example.com">Click Here</
a><script>alert("XSS")</script>'
safe_string = sanitize_with_bleach(unsafe_string)
print()
print(unsafe_string)
print()
print(safe_string)
```

Web App with Javascript and Python

This basic Bottle web app shows you how you can use Javascript and Python together to protect your app.

Javascript checks that name, age and email are the right data types before sending form data to Python. Python can then use html.escape() to escape HTML tags in the Name field.

webapp-clean.py

```
from bottle import run, route, post, request
import html

@route('/')
def index():
    page = '''
            <h1>Hello</h1>

            <form action="process" method="post" id="form">
                Name: <input type="text" name="name" id="name" 
required><br><br>
                Age: <input type="number" name="age" id="age" 
required><br><br>
                Email: <input type="email" name="email" id="email" 
required><br><br>
                <input type="submit" value="Submit" 
onclick="validate(event)">
            </form>

            <script>
                function validate(event) {
                    event.preventDefault();

                    const name = document.getElementById("name");
                    const age = document.getElementById("age");
                    const email = document.getElementById("email");

                let message = "";

                if (!name.validity.valid) {
                    message += "Name is not valid\\n";
                }

                if (!age.validity.valid) {
                    message += "Age is not valid\\n";
                }

                if (!email.validity.valid) {
                    message += "Email is not valid\\n";
                }

                if (message) {
                    alert(message);
                } else {
                    document.getElementById("form").submit();
}
            }
            </script>
            '''
    return page

@post('/process')
def process():
    name = request.forms.get('name')
    age = request.forms.get('age')
    email = request.forms.get('email')

    #name = html.escape(name)

    page = f'''
            <h1>Name: {name}</h1>
            <h1>Age: {age}</h1>
            <h1>Email: {email}</h1>
            '''
    return page

run(host='0.0.0.0', port='8080')
```

Final Thoughts - RegEx...

RegEx makes this much easier....