CS1116/CS5018

Web Development 2

Dr Derek Bridge

School of Computer Science & Information Technology
University College Cork

JavaScript preliminaries

- Suppose your Web page contains a textfield:

```
<input type="text" name="my_textfield" id="my_textfield">
```

- Suppose your JavaScript retrieves it:

```
let element = document.querySelector("#my_textfield");
```

- To change what is displayed in the textfield:

```
element.value = "Hello";
```

- To access what the user has typed into the textfield:

```
let users_input = element.value;
```

- This is another example of changing/accessing attribute values — lecture 14

Version 1

- A program that converts your height from feet-and-inches to centimetres
- Version 1 comprises:
  - `heightv1.py`

  — a self-processing page
- Validation of user input — done server-side by `heightv1.py`
- Calculation of height in cm — done server-side by `heightv1.py`

heightv1.py

```
#!/usr/local/bin/python3

from cgitb import enable
enable()

from cgi import FieldStorage

print('Content-Type: text/html')
print()

def check_for_int(text, minimum, maximum):
    if text.strip() == '':
        return 'Required'
    try:
        number = int(text)
        if number < minimum:
            return 'Must be no less than ' + str(minimum)
        if number > maximum:
            return 'Must be no greater than ' + str(maximum)
        return ''
    except ValueError:
        return 'Must be a whole number'

form_data = FieldStorage()
feet_input = '0'
feet_int = 0
feet_msg = ''
inches_input = '0'
inches_int = 0
inches_msg = ''
cm_int = 0
if len(form_data) != 0:
    feet_input = form_data.getfirst('feet_input', '')
    inches_input = form_data.getfirst('inches_input', '')
    feet_msg = check_for_int(feet_input, 0, 8)
    inches_msg = check_for_int(inches_input, 0, 11)
    if not feet_msg and not inches_msg:
        feet_int = int(feet_input)
        inches_int = int(inches_input)
        cm_int = (12 * feet_int + inches_int) * 2.5
print("""
    <!DOCTYPE html>
    <html lang="en">
        <head>
            <meta charset="utf-8" />
            <title>Getting high</title>
        </head>
        <body>
            <p>
                How tall are you?
            </p>
            <form action="heightv1.py" method="get">
                <label for="feet_input">Feet: </label>
                <input type="text" name="feet_input" id="feet_input" value="%s" />
                <span id="feet_msg">%s</span>
                <label for="inches_input">Inches: </label>
                <input type="text" name="inches_input" id="inches_input" value="%s" />
                <span id="inches_msg">%s</span>
                <label for="cm_input">Centimetres: </label>
                <input type="text" name="cm_input" id="cm_input" value="%i" disabled />
                <input type="submit" />
```

```
            </form>
        </body>
    </html>""" % (feet_input, feet_msg, inches_input, inches_msg, cm_int))
```

Version 2

- Version 2 comprises:
  - `heightv2.py`
  - `heightv2.js`
- Validation of user input:
  - done client-side by `heightv2.js` when user submits
  - but also done server-side by `heightv2.py`

  Q: What's the advantage of doing it client-side?
  Q: Why the duplication? Why do we need to do it server-side as well?
- Calculation of height in cm — done server-side by `heightv2.py`

heightv2.py

Identical to `heightv1.py` except:

```
print("""
    <!DOCTYPE html>
    <html lang="en">
        <head>
            <meta charset="utf-8" />
            <title>Getting high</title>
            <script src="heightv2.js" type="module"></script>
        </head>
        <body>
            <p>
                How tall are you?
            </p>
            <form action="heightv2.py" method="get">
                <label for="feet_input">Feet: </label>
                <input type="text" name="feet_input" id="feet_input" value="%s" />
                <span id="feet_msg">%s</span>
                <label for="inches_input">Inches: </label>
                <input type="text" name="inches_input" id="inches_input" value="%s" />
                <span id="inches_msg">%s</span>
                <label for="cm_input">Centimetres: </label>
                <input type="text" name="cm_input" id="cm_input" value="%i" disabled />
                <input type="submit" />
            </form>
        </body>
    </html>""" % (feet_input, feet_msg, inches_input, inches_msg, cm_int))
```

heightv2.js

```
let form_element;
let feet_input;
let feet_span;
let inches_input;
let inches_span;
let cm_input;

document.addEventListener('DOMContentLoaded', init, false);

function init() {
    feet_input = document.querySelector('#feet_input');
    feet_span = document.querySelector('#feet_msg');
    inches_input = document.querySelector('#inches_input');
    inches_span = document.querySelector('#inches_msg');
    cm_input = document.querySelector('#cm_input');
    form_element = document.querySelector('form');
    form_element.addEventListener('submit', validate_input, false);
}

function check_for_int(text, minimum, maximum) {
    let trimmed_text = text.trim();
    if (trimmed_text === "") {
        return "Required";
    }
    let number = ~~Number(trimmed_text);
    if (String(number) !== trimmed_text) {
        return "Must be a whole number";
    }
    if (number < minimum) {
        return "Must be no less than " + minimum;
    }
    if (number > maximum) {
        return "Must be no greater than " + maximum;
    }
    return '';
}

function validate_input(event) {
    feet_msg = check_for_int(feet_input.value, 0, 8);
    inches_msg = check_for_int(inches_input.value, 0, 11);
    feet_span.innerHTML = feet_msg;
    inches_span.innerHTML = inches_msg;
    if (feet_msg || inches_msg) {
        event.preventDefault();
    }
}
```

Notes on Version 2

- The form listens for submit events:

```
form_element.addEventListener('submit', validate_input, false);
```

- The listener function looks like this:

```
function validate_input(event) {
    …
    if (feet_msg || inches_msg) {
        event.preventDefault();
    }
}
```

Q: What is the purpose of the last statement in the function?

Version 3

- Version 3 comprises:
  - `heightv3.py`
  - `heightv3.js`
- Identical to Version 2 except that client-side validation of user input is done:
  - whenever user changes what s/he has typed and
  - when s/he submits (as in Version 2)

`heightv3.js`

Identical to `heightv2.js`, except:

```
function init() {
    feet_input = document.querySelector('#feet_input');
    feet_span = document.querySelector('#feet_msg');
    inches_input = document.querySelector('#inches_input');
    inches_span = document.querySelector('#inches_msg');
    cm_input = document.querySelector('#cm_input');
    form_element = document.querySelector('form');
    form_element.addEventListener('submit', validate_input, false);
    feet_input.addEventListener('change', validate_input, false);
    inches_input.addEventListener('change', validate_input, false);
}
```

Version 4

- We don't need the server-side Python program! JavaScript can do the calculations client-side
- Version 4 comprises:
  - `heightv4.html`
  - `heightv4.js`
- Validation of user input — done client-side by `heightv4.js`
- Calculation of height in cm — done client-side by `heightv4.js`

heightv4.html

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>Getting high</title>
        <script src="heightv4.js" type="module"></script>
    </head>
    <body>
        <p>
            How tall are you?
        </p>
        <form>
            <label for="feet_input">Feet: </label>
            <input type="text" name="feet_input" id="feet_input" value="0" />
            <span id="feet_msg"></span>
            <label for="inches_input">Inches: </label>
            <input type="text" name="inches_input" id="inches_input" value="0" />
            <span id="inches_msg"></span>
            <label for="cm_input">Centimetres: </label>
            <input type="text" name="cm_input" id="cm_input" value="0" disabled />
            <input type="submit" />
        </form>
    </body>
</html>
```

heightv4.js

```javascript
let form_element;
let feet_input;
let feet_span;
let inches_input;
let inches_span;
let cm_input;

document.addEventListener('DOMContentLoaded', init, false);

function init() {
    feet_input = document.querySelector('#feet_input');
    feet_span = document.querySelector('#feet_msg');
    inches_input = document.querySelector('#inches_input');
    inches_span = document.querySelector('#inches_msg');
    cm_input = document.querySelector('#cm_input');
    form_element = document.querySelector('form');
    form_element.addEventListener('submit', convert_to_cm, false);
    feet_input.addEventListener('change', validate_input, false);
    inches_input.addEventListener('change', validate_input, false);
}

function check_for_int(text, minimum, maximum) {
    let trimmed_text = text.trim();
    if (trimmed_text === "") {
        return "Required";
    }
    let number = ~~Number(trimmed_text);
    if (String(number) !== trimmed_text) {
        return "Must be a whole number";
    }
    if (number < minimum) {
        return "Must be no less than " + minimum;
    }
    if (number > maximum) {
        return "Must be no greater than " + maximum;
    }
    return '';
}

function validate_input(event) {
    feet_msg = check_for_int(feet_input.value, 0, 8);
    inches_msg = check_for_int(inches_input.value, 0, 11);
    feet_span.innerHTML = feet_msg;
    inches_span.innerHTML = inches_msg;
}

function convert_to_cm(event) {
    feet_msg = check_for_int(feet_input.value, 0, 8);
    inches_msg = check_for_int(inches_input.value, 0, 11);
    feet_span.innerHTML = feet_msg;
    inches_span.innerHTML = inches_msg;
    if (! feet_msg && ! inches_msg) {
        feet_int = Number(feet_input.value);
        inches_int = Number(inches_input.value);
        cm_int = (12 * feet_int + inches_int) * 2.5;
        cm_input.value = cm_int;
    }
    event.preventDefault();
}
```

Notes on Version 4

- Now, everything is being done client-side — using JavaScript
- Q: What are the cost-benefit trade-offs here?
- Q: When *must* we use a server-side approach?