W4D4

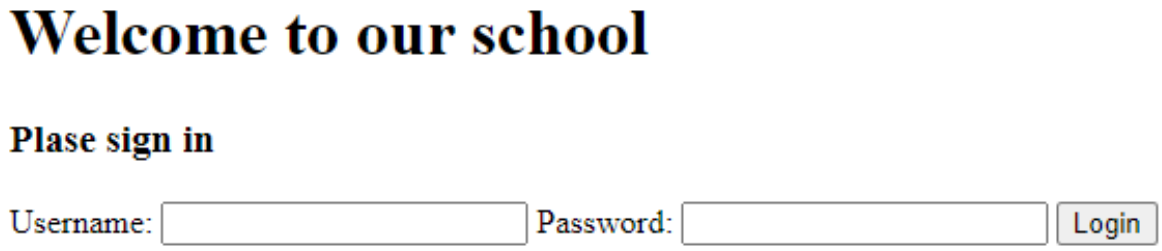
Asynchronous Javascript

HTML: Forms

REVATURE

Used to collect data from the user

- <form>
  - Attributes
    - action
    - target
    - method
- <input>
  - Attributes
    - name
    - value
    - placeholder
    - required
    - min/max




```
<body>
    <h1>Welcome to our school</h1>
    <h3>Plase sign in</h3>

    <!-- You can include the method and action attributes to tell the form
            where to send information and how to send it
        -->
    <form>
        <label for='username'>Username: </label>
        <input type='text' name='username' maxlength='50' required />
        <label for='password'>Password: </label>
        <input type='password' name='password' minlength='8' required />
        <input type='submit' value='Login' />
    </form>
</body>
```

HTML: Input Element and Types

REVATURE

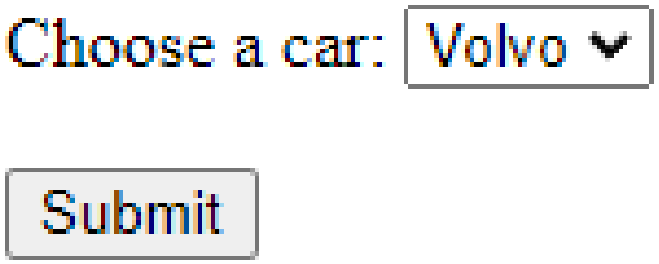
Input element collects the form data

- Input types
  - text
  - password
  - radio
  - check boxes
  - file select
  - textarea
  - select
  - submit
  - reset

HTML: Select and Multi-Select

- Special input field which allows for a drop down of values
  - Special attribute called `multiple` allows users to choose multiple drop-down choices

```
<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select name="cars" id="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="opel">Opel</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit" value="Submit">
</form>
```



HTML 5: Validation

REVATURE

Make sure the data is correct on the client side

- With javascript, or form validation
- HTML5 validation techniques
  - required
  - minlength or maxlength
  - min or max
  - type
  - pattern

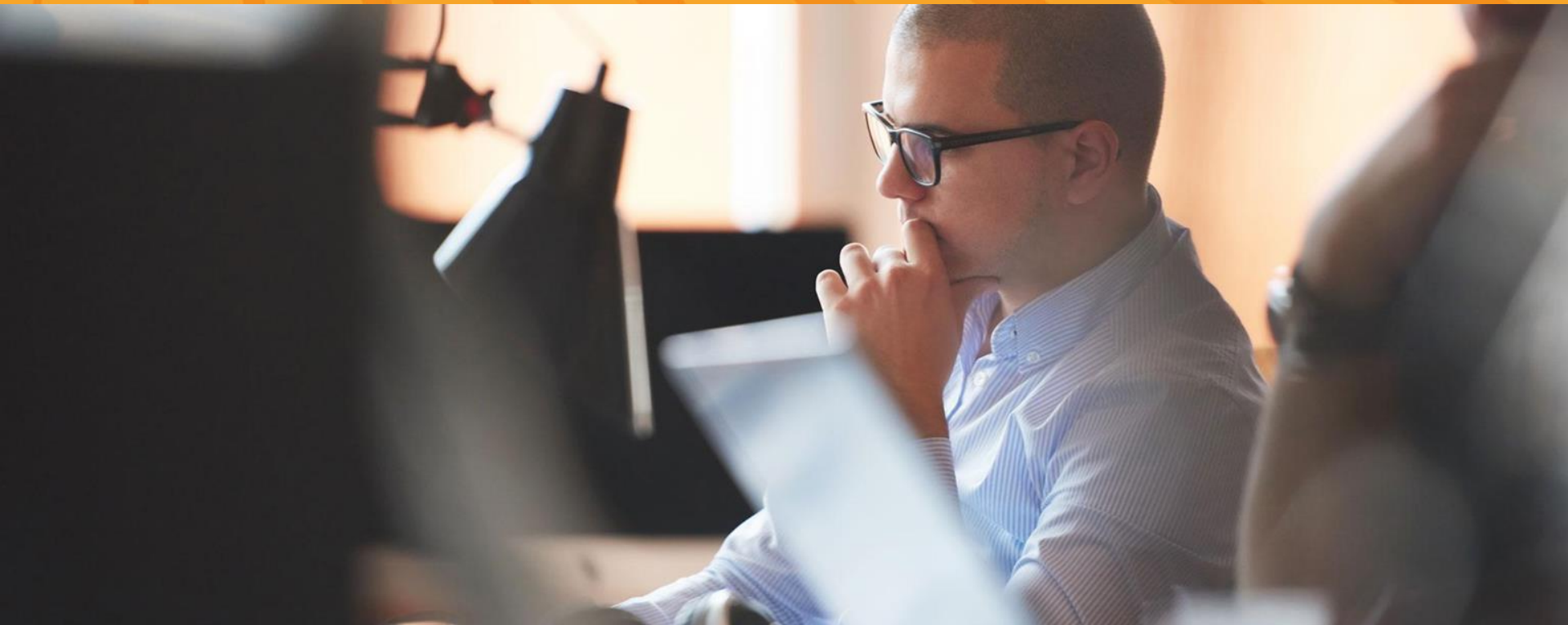
Javascript: Submitting Forms

REVATURE

1. Using default form action
   - Sends the information directly to the url provided
   - Automatically reloads the page
2. Register an event listener
   - When a user submits, the submit listener prevents default
   - Allows users to transform/verify data
   - Prevents page reloading on bad input
   - Send the http request manually

REVATURE

HTML Form Demo



Asynchronous Programming

REVATURE

Asynchronous Code:

- No need to wait for a task to finish
- Start a function and move onto other code
- Useful for:
  - Lengthy functions
  - API calls
  - Processes that block the main execution

Asynchronous Javascript

REVATURE

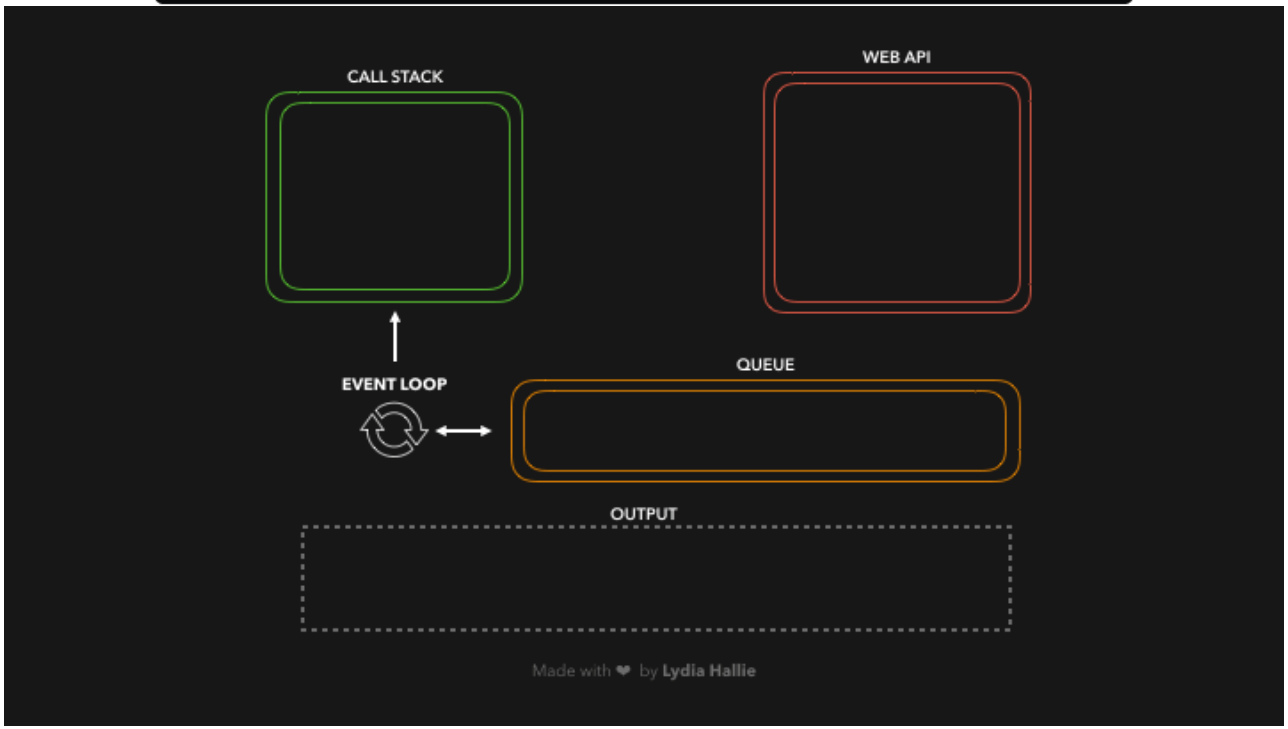
Javascript is single threaded

- Method calls are added to a stack
- Each method gets called one at a time
- Browser has extra threads to be used
- JS implements a queue of functions processed by the browser's threads
- This queue is used by the event loop

Javascript: Event Loop

```
const foo = () => console.log("First");
const bar = () => setTimeout(() => console.log("Second"), 500);
const baz = () => console.log("Third");

bar();
foo();
baz();
```



Source: https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

Javascript: Event Loop Steps

REVATURE

1. Asynchronous call gets added to the stack
2. JS decides to hand it off to the web api where it processes
3. Other functions can be added and executed on the call stack while the web api works
4. Once the web api has a response it gets added to the queue
5. The event loop checks the queue when the call stack is empty to see if something needs to be added to the call stack
6. Our response/callback is added to call stack and process as normal

Javascript: Promises

REVATURE

Used as placeholders for future data

- Objects which use callback function called executor which runs when created
  - Executor calls resolve when the data is there
  - Executor calls reject when an error occurs
- Status property gives us info on the state
  - Pending
  - Fulfilled
  - Rejected

Javascript: Handling Promises

REVATURE

Promises use consuming functions to handle when a promise is successful for results in an error

- .then() is used to consume the result
- .catch() is used to handle an error
- .finally() is ran no matter what
- You can chain together as many .then() as you like

Javascript: Errors

REVATURE

Objects created when something goes wrong

- Runtime: Error occurring as a result of code
- User Defined: Custom error thrown by the developer
- Handle Errors with try/catch blocks like Java
- Throw Errors with the `throw` keyword like Java
  - Unlike java, any object can be thrown in Javascript

Javascript: Custom Errors

REVATURE

- Errors are objects like everything else in Javascript
- You can create a custom Error object two ways:
- Create a function named as an Error
  - set the __prototype__ property to Error
- Create a class which extends Error
  - Use the super() keyword to pass an error massage

Javascript: Fetch API

REVATURE

More modern and versatile way of making AJAX Requests

- Built into the browser, uses .fetch() method
- Returns a promise that is retrieved from the response
  - Successful request resolves
  - error response is rejected

Methods to access the response body:

- response.text()
- response.json()
- response.formData()
- response.blob()
- response.arrayBuffer()

REVATURE

Async Javascript Demo

Javascript: async/await

REVATURE

ES7 feature which simplifies async javascript

- Use the async keyword to denote a function runs asynchronously
  - Implicitly returns a promise
- Use await keyword if you explicitly want to wait for the promise to resolve/reject
  - Only available in an async function
  - The program can still continue, but the function does not finish until the wait is over

Javascript: Timing Events

REVATURE

- Functions attached to the global window which allow us to automate or run tasks after a specified amount of time
- setTimeout()
  - Execute a callback after x amount of milliseconds
  - .clearTimeout() is used to cancel a timeout
- setInterval()
  - Execute a callback once every x amount of milliseconds
  - .clearInterval() to stop the interval

REVATURE

Fetch with async/await Demo