JavaScript

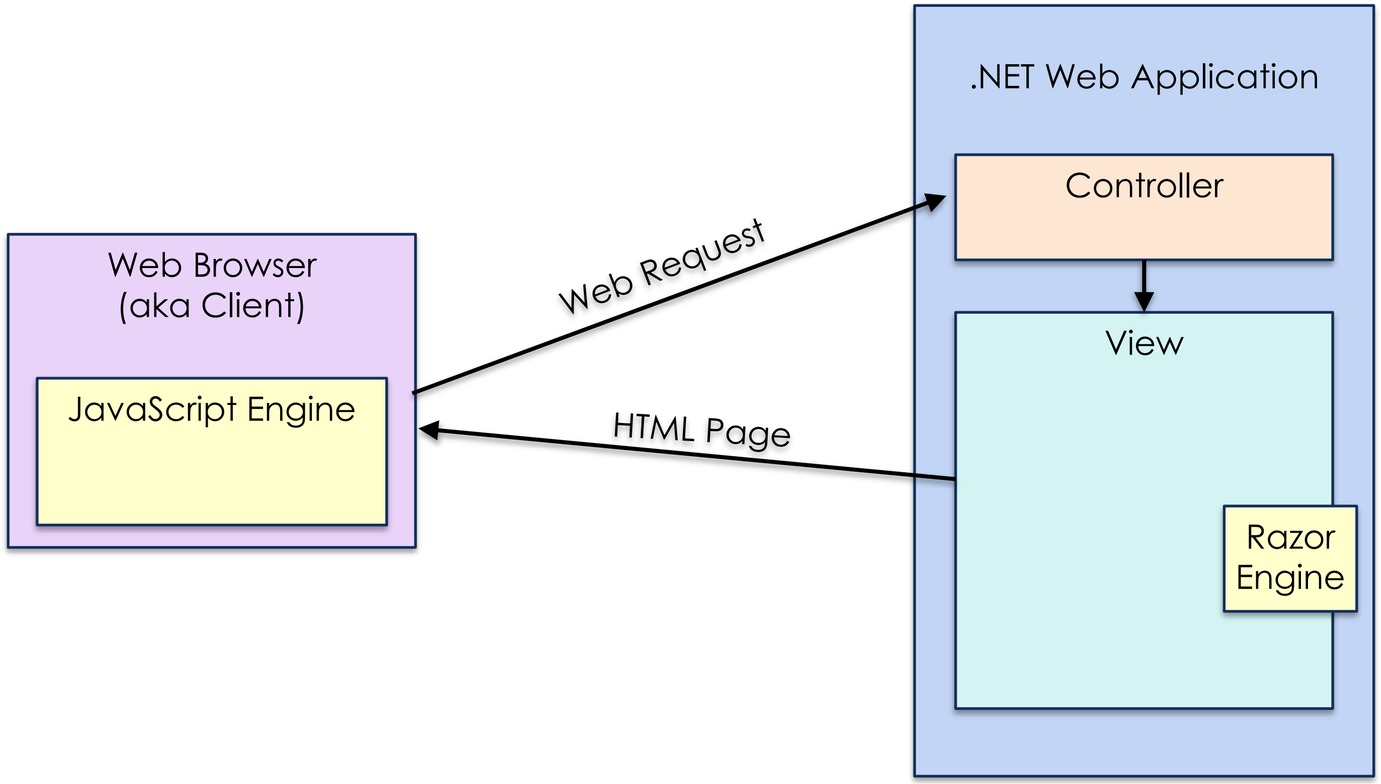
Tan Cher Wah (cherwah@nus.edu.sg)

JavaScript

- JavaScript is a programming language that is used to add interactivity to web pages
- Using JavaScript code, a web page can respond to user input and update its content accordingly, without requiring a full page reload
- It started off as a Client-side language, which means it runs in the user's web browser
- JavaScript is supported by all major web browsers (e.g. Chrome, Edge, Firefox)

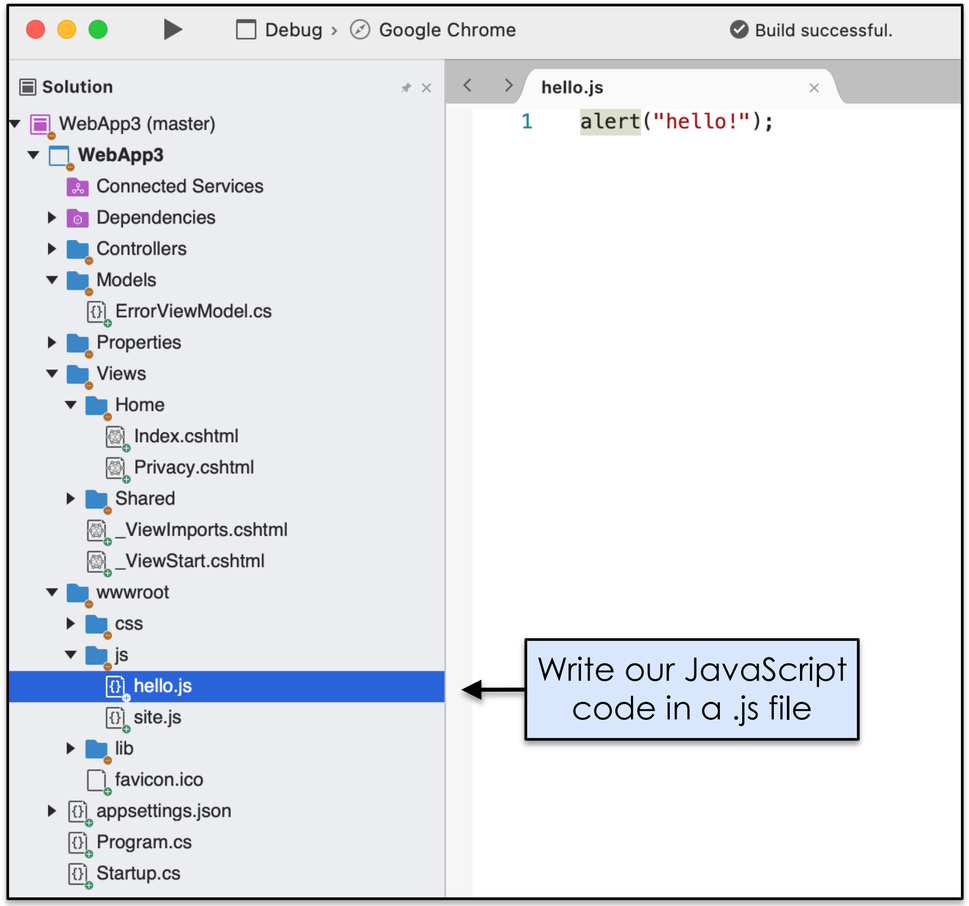
JavaScript in a .NET project

- .NET MVC is a server-side technology and does not support user interactivity on the client-side (e.g. browser)
- JavaScript, is used in a .NET project, to provide client-side interactivity



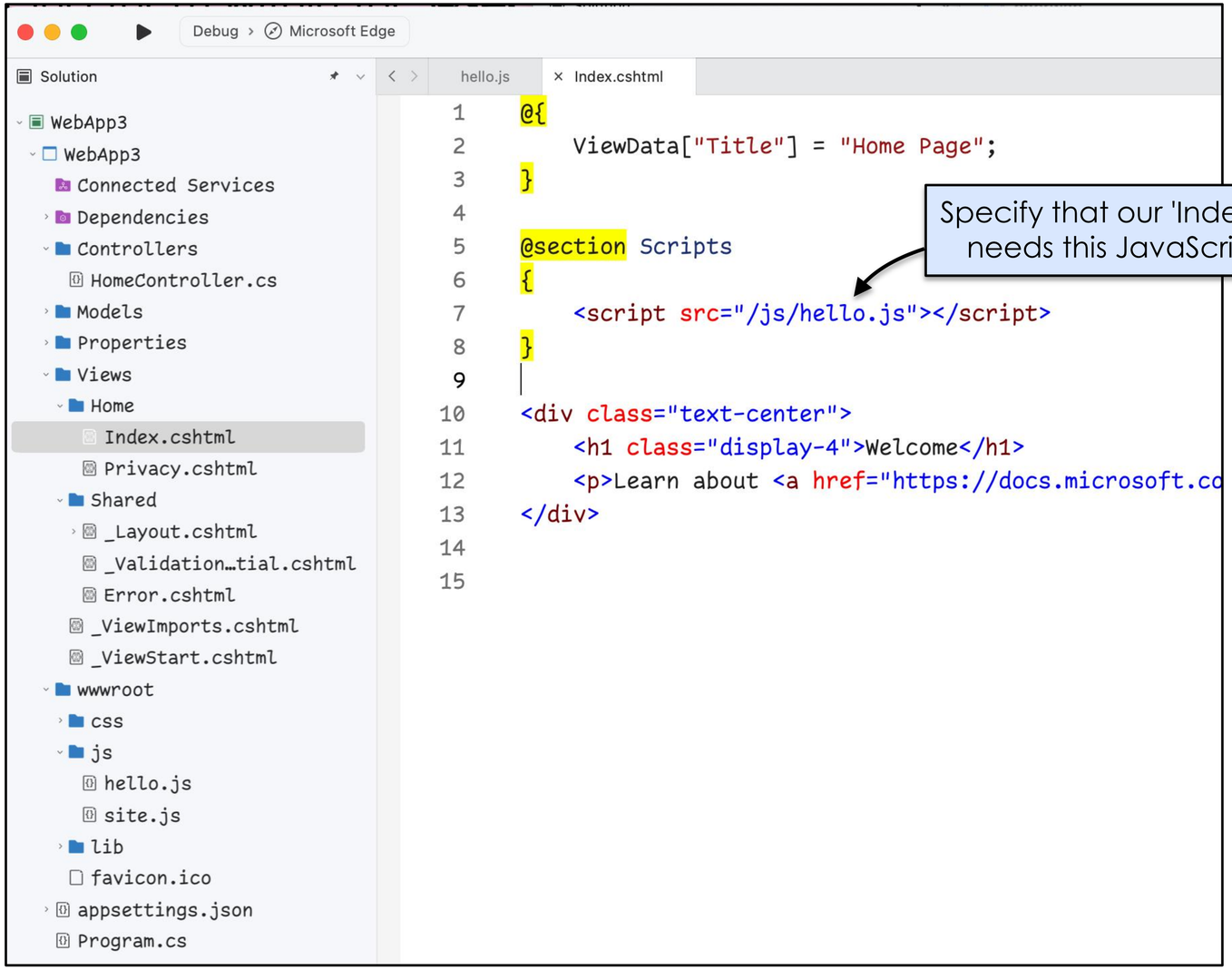
JavaScript files in a .NET project

Place your JavaScript files in the wwwroot/js folder of your project; common or shared JavaScript code can be stored in the wwwroot/js/site.js file



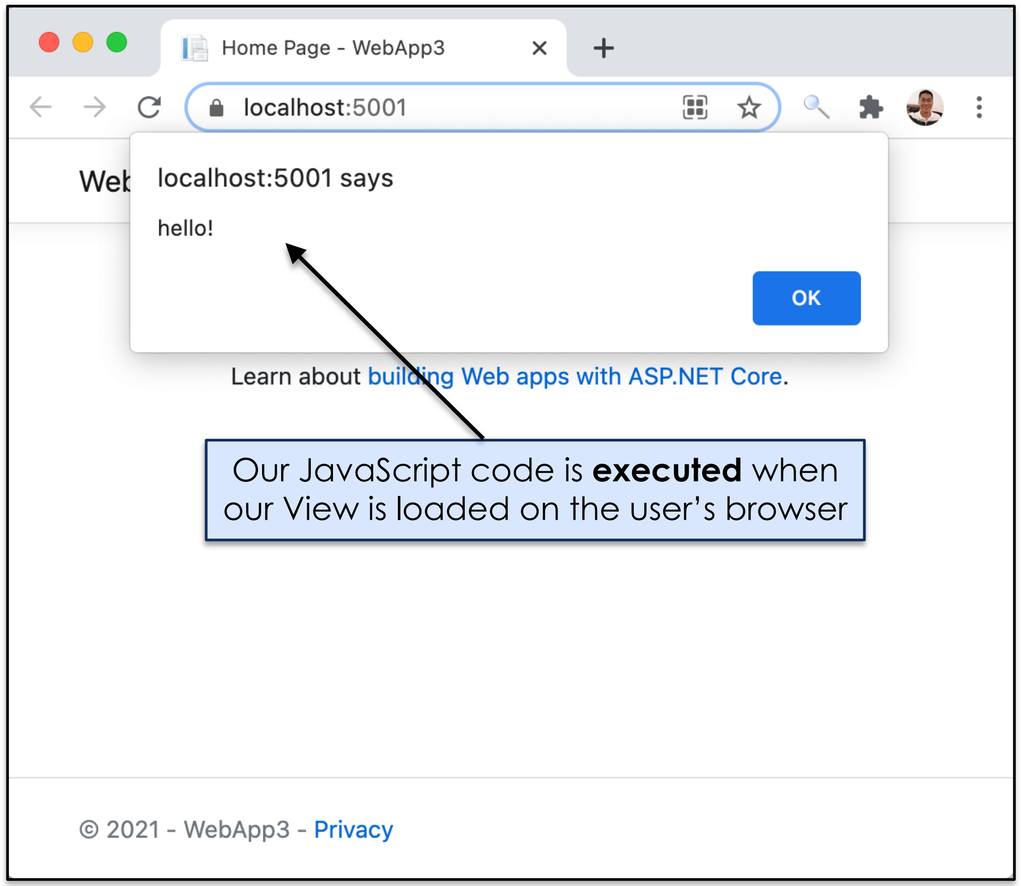
Pairing JavaScript with our View

Pair JavaScript file(s) with a View by placing them within the @section Scripts of that View



Running JavaScript

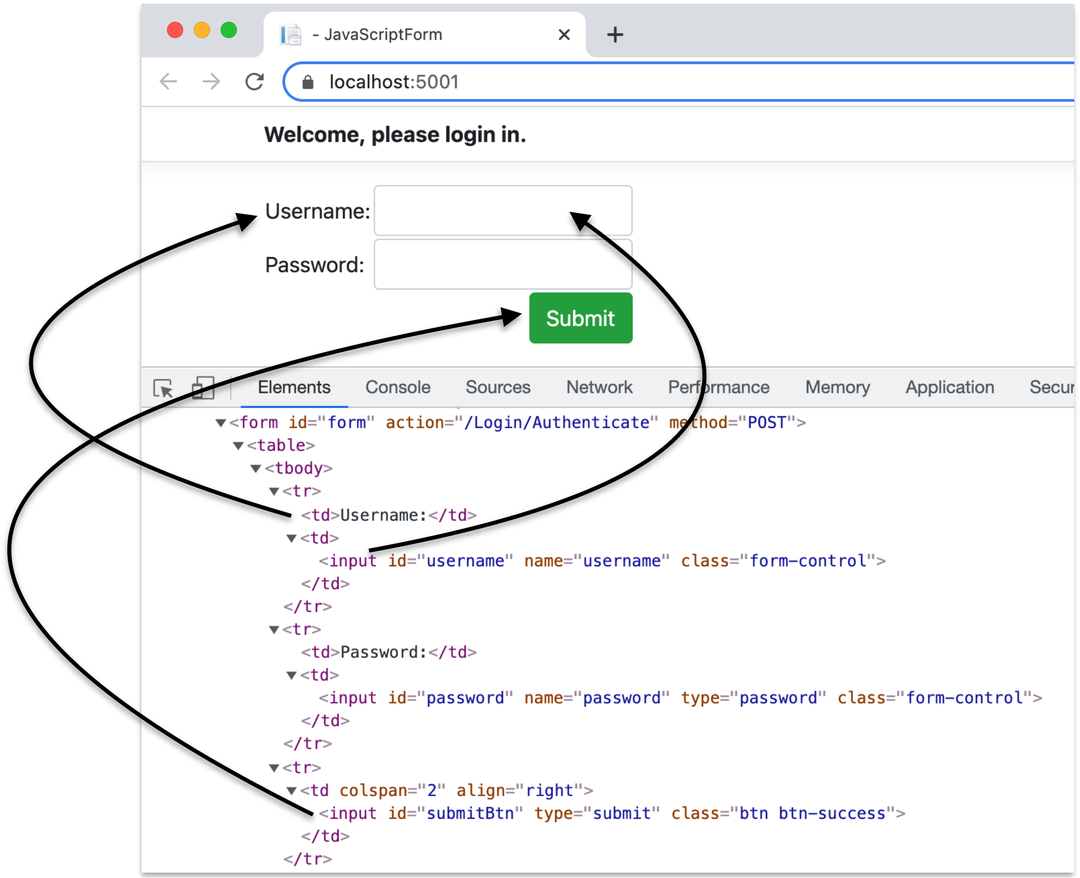
A web browser is responsible for fetching and loading our Javascript code; and, upon loading, executes any JavaScript code outside a function



DOCUMENT OBJECT MODEL

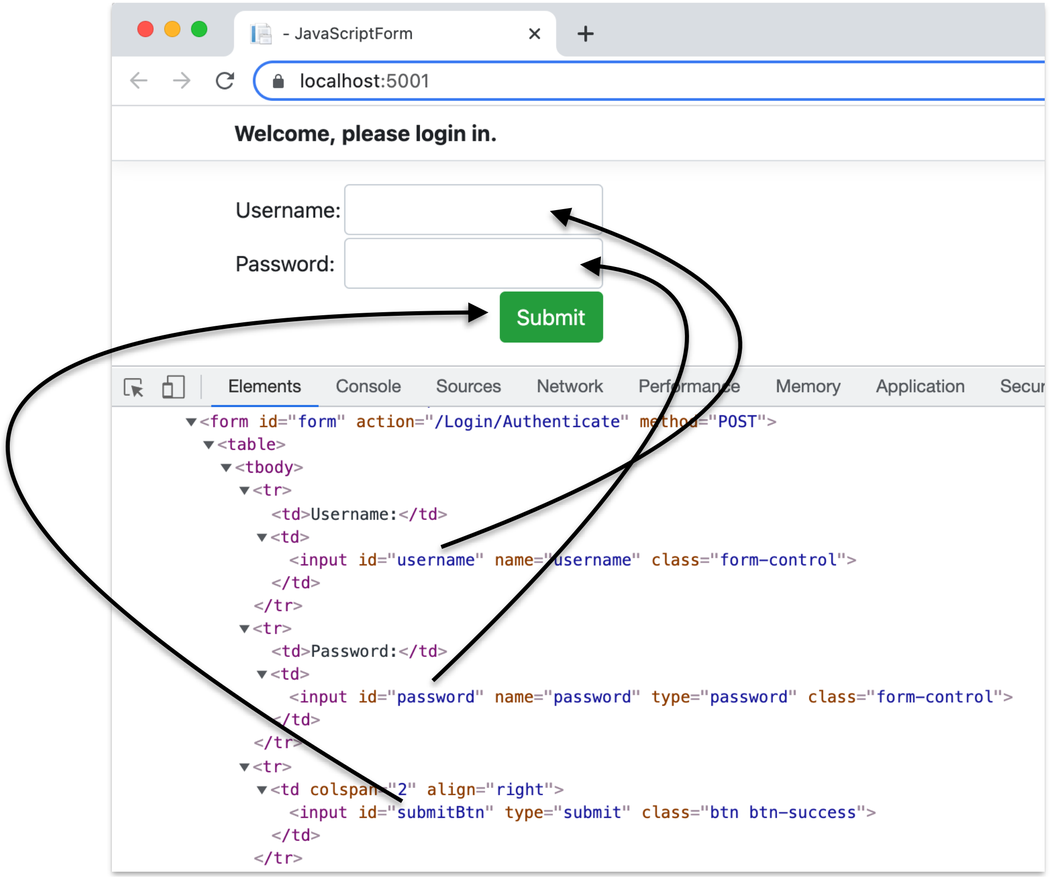
Concept of DOM

A Document Object Model (DOM) represents a tree structure of a HTML document



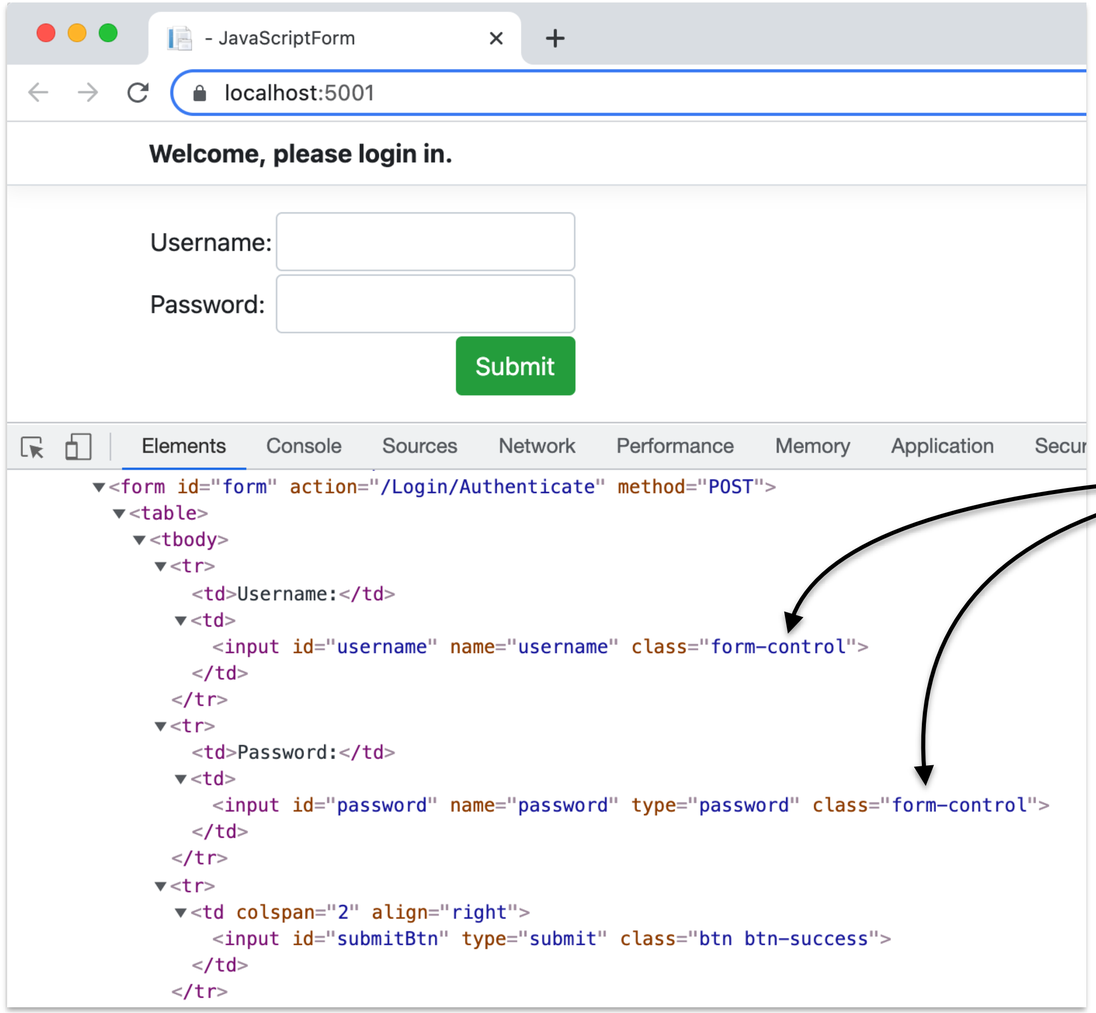
DOM – The concept of an ‘id’

An 'id' attribute refers to a single HTML element and is unique within a HTML document



DOM – The concept of a ‘class’

A 'class' attribute refers to a class of one or more HTML elements in a HTML document



Same styling (e.g. “form-control”) applied to both textboxes

Styling-values such as “form-control” and “btn btn-success” are defined in Bootstrap’s stylesheet

Styles can be stacked – for example “btn btn-success” applies two styles on the button element

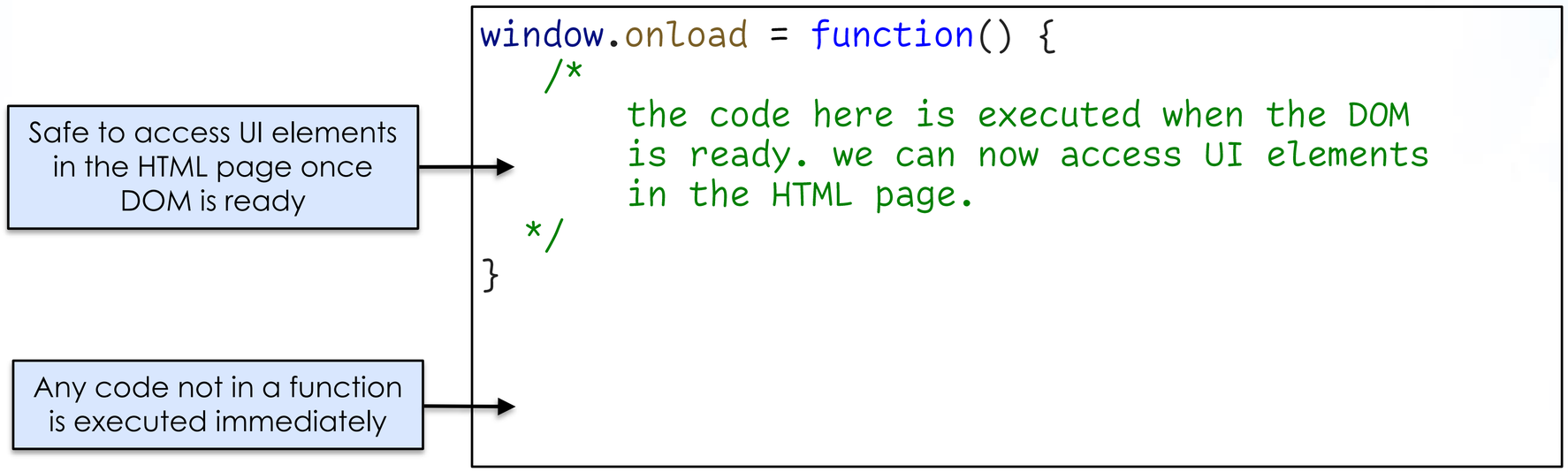
Uses for 'id' and 'class'

- Interact with tagged elements
  - Hide or Show an element (e.g. button)
  - Fetch values from an element (e.g. input box)
- Style tagged elements
  - Add or remove borders to an element (e.g. table)
  - Change the font-color of an element (e.g. textbox)
- Observe tagged elements
  - Detect mouse-clicks for an element (e.g. button)
  - Detect keyboard focus has left an element (e.g. textbox)

CODING WITH JAVASCRIPT

Wait for DOM to be ready

- The DOM takes time to be constructed when a HTML document is first loaded into a browser
- HTML elements are not accessible before the DOM is ready
- window.onload() is called, by the JavaScript engine, when DOM is ready



Get an element via its ID

Use document.getElementById(<your_id_name>) to get a single HTML element with id=<your_id_name>

```html
<html>
<body>

<input id="username" />
<input id="password" />

</body>
</html>
```

HTML

```javascript
// get the HTML textbox-elements
let username_elem = document.getElementById("username");
let password_elem = document.getElementById("password");

// get the values in the text boxes
let username = username_elem.value;
let password = password_elem.value;
```

JavaScript

Get elements via their Class

Use document.getElementsByClassName(<your_class_name>) returns a list of HTML elements with class=<your_class_name>

```html
<html>
<body>

<div class="my_clickable">Title 1</div>
<div class="my_clickable">Title 2</div>
<div class="my_clickable">Title 3</div>

</body>
</html>
```

HTML

```javascript
let elems = document.getElementsByClassName("my_clickable");

for (let i = 0; i < elems.length; i++) {
  // an element tagged with "class=my_clickable"
  let elem = elems[i];
    ...
}
```

JavaScript

DIV

A DIV tag is a generic container; use innerHTML to get or set its content

```html
<html>
<body>

<div id="msg"></div>

</body>
</html>
```

HTML

```javascript
let msg_elem = document.getElementById("msg");

/* get existing message */
let old_msg = msg_elem.innerHTML;

/* set new message */
msg_elem.innerHTML = "My New Message";
```

JavaScript

Adding Event Listeners

Use addEventListener(<event>, <callback_function>) to register events, such as mouse-clicks, for HTML elements

```html
<html>
<body>

<div class="my_clickable">Title 1</div>
<div class="my_clickable">Title 2</div>
<div class="my_clickable">Title 3</div>

</body>
</html>
```

HTML

```javascript
let elems = document.getElementsByClassName("my_clickable");

for (let i = 0; i < elems.length; I++) {
  let elem = elems[i];

  /* add a mouse-click event listener */
  elem.addEventListener("click", OnClick);
}

function OnClick(event) {
  // element registered with the click-event
  let elem = event.currentTarget;
}
```

JavaScript

Custom Attributes

- Associate application's data with a HTML element via custom attributes
- Server-side (of a web-application) usually embed custom attributes so that the Client-side (JavaScript logic) has more information to work with

HTML

```
<html>
<body>

<div class="my_clickable" timetoread="3" start="1">Title 1</div>
<div class="my_clickable" timetoread="5" start="3">Title 2</div>
<div class="my_clickable" timetoread="4" start="6">Title 3</div>

</body>
</html>
```

JavaScript

```
let elems = document.getElementsByClassName("my_clickable");

for (let i = 0; i < elems.length; I++)
{
   let elem = elems[i];

   /* get custom data embedded in each element */
   let read_time = elem.getAttribute("timetoread");
   let start_pg = elem.getAttribute("start")
}
```

Styling elements via StyleSheets

The DIV element (left) is being styled via CSS (right)

```html
<html>
<body>

<div id="content" class="hilite">My Content</div>

</body>
</html>
```

HTML

```css
.hilite {
  border: 3px solid #5499c7;
  background-color: #f1f7fc;
}
```

Style-Sheet

Styling elements via StyleSheets

A style can be added or removed dynamically using JavaScript

```html
<html>
<body>

<div id="content">My Content</div>

</body>
</html>
```

HTML

```javascript
let elem = document.getElementById("content");

// toggle clicked element
if (! elem.classList.contains("hilite"))
{
  elem.classList.add("hilite");
}
else
{
  elem.classList.remove(“hilite");
}
```

JavaScript

Styling elements via JavaScript

The visibility of the content (left) is updated programmatically via the JavaScript (right)

```html
<html>
<body>

<div id="content">My Content</div>

</body>
</html>
```

HTML

```javascript
let elem = document.getElementById("content");

// toggle display of an element
if (elem.style.display == "none")
{
  elem.style.display = "display";
}
else
{
  elem.style.display = "none";
}
```

JavaScript

HTML FORM

HTML Form

A HTML Form is used for accepting inputs from the user

The action is set to the receiving Controller and Action Method of our .NET Core application

The method should be POST for submitting a form

The **id** values are used by our JavaScript code

The **name** values are used by our Server code

```
<form id="form" action="/Login/Authenticate" method="POST">
    <table>
        <tr>
            <td>Username:</td>
            <td>
                <input id="username" name="username" class="form-control" />
            </td>
        </tr>
        ...
        <tr>
            <td colspan="2" align="right">
                <input id="submitBtn" type="submit" class="btn btn-success" />
            </td>
        </tr>
    </table>
</form>
```

For forms, Set <input> tag as type="submit"

The web browser will style this UI element as a button

Validating a Form

A HTML Form can be validated with JavaScript before user's inputs are sent back to our web application

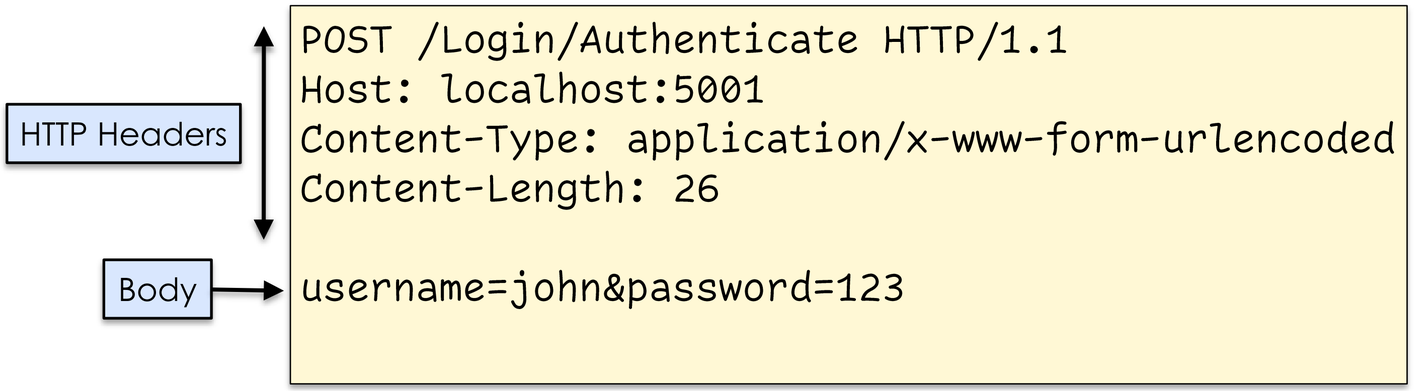
onsubmit() is called when user clicks on the Submit button to submit the form

```
window.onload = function()
{
    let form = document.getElementById("form");

    form.onsubmit = function(event) {
      let username_elem = document.getElementById("username");

      let username = "";
      if (username_elem) {
        username = username_elem.value.trim();
      }

      if (username.length == 0) {
        // cancel form post
        event.preventDefault();
      }

      // go ahead and post form
    }
}
```

To cancel form submission

Under the Hood

- In HTTP, the default encoding for posting a Form to a web server is application/x-www-form-urlencoded
- Form data are encoded as key-value pairs before sending, like so: `<key1>=<value1>&<key2>=<value2>&<key3>=<value3>`
- Over the wire, this is what is being sent:



When the Form Post reaches the Server

To receive the values ‘john’ and ‘123’, specify matching names in parameters of our action method of our Controller

```
public IActionResult Authenticate(string username, string password)
{
  bool success = false;

  /* perform some checks first */

  if (success)
  {
    // checks succeeded, brings user to landing page
    return RedirectToAction("Index", "Home");
  }

  // checks failed, returns user to login page
  return RedirectToAction("Index", "Login");
}
```

The parameter names must match with the names specified in the form

IFormCollection

Alternatively, use IFormCollection (which functions like a Dictionary) to retrieve key-value pairs from the Form Post

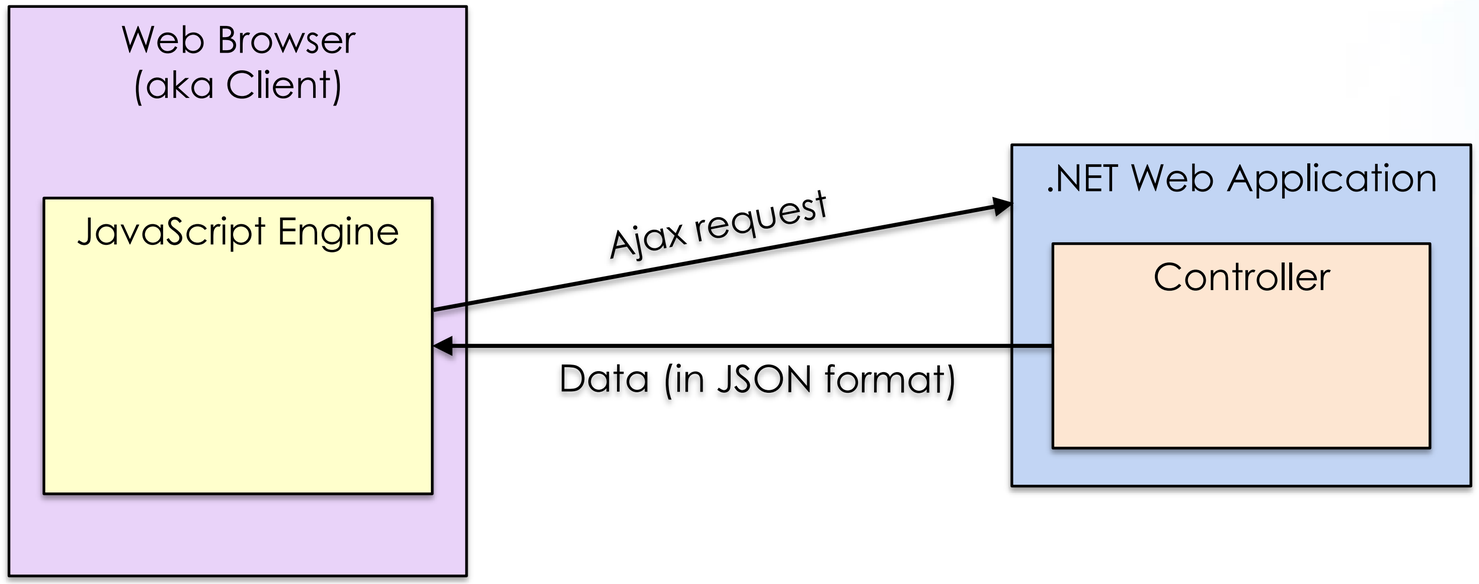
```
public IActionResult Authenticate(IFormCollection form)
{
  bool success = false;

  /* the keys must match the "name" values of the form */
  string username = form["username"];
  string password = form["password"];

  /* perform checks here */

  if (success) {
    // checks succeeded, brings user to landing page
    return RedirectToAction("Index", "Home");
  }

  // checks failed, returns user to login page
  return RedirectToAction("Index", "Login");
}
```

Key-Value pairs retrieval

AJAX

AJAX

AJAX stands for Asynchronous JavaScript And XML, and is a technique to send or receive data to/from our web application, without the need to request for a new HTML page



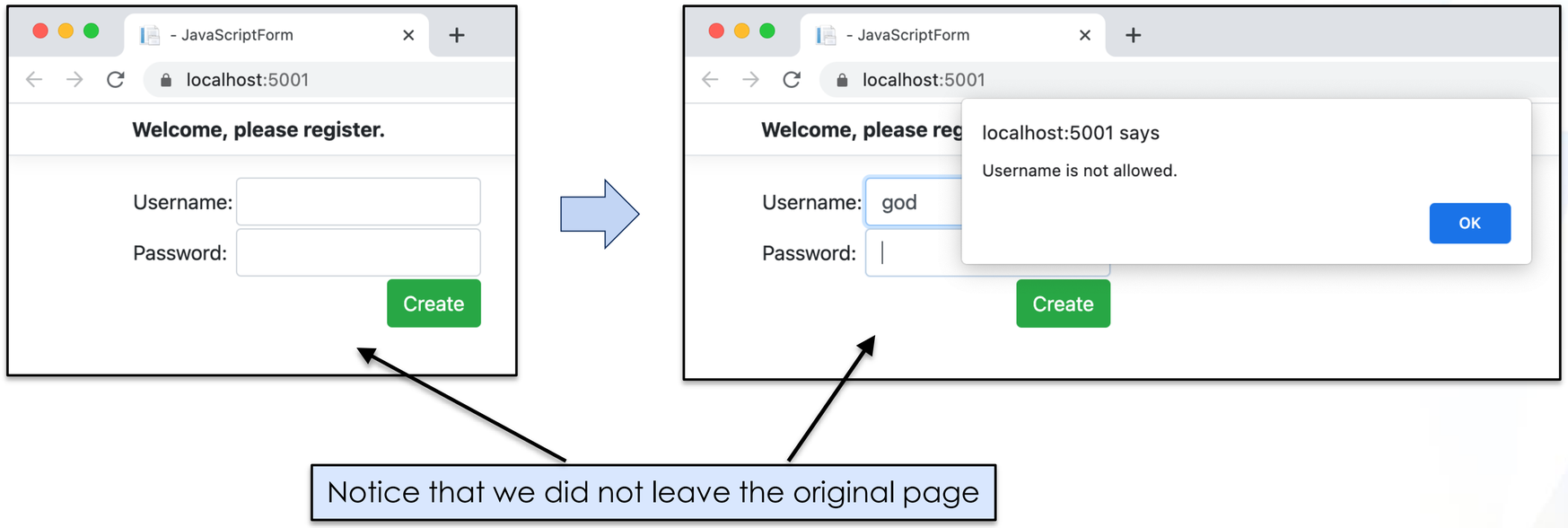
JSON

- JSON is a common data transfer format for AJAX
- A JSON object starts off with a ‘{‘ and ends with a ‘}’
- { } denotes an object/dictionary
- [ ] denotes an array
- Key-Value pairs
- Key must be a *string;* Value can be a *array, string, integer, float, boolean, null or user-defined object*

```
{
  "books": [
    {
      "id": 555,
      "title": "The C Programming Language",
      "authors": [
        "Dennis Ritchie",
        "Brian Kernighan"
      ],
      "activity_royalty": false
    },
    {
      "id": 556,
      "title": "The C++ Programming Language",
      "authors": [
        "Bjarne Stroustrup"
      ],
      "active_royalty": true
    }
  ]
}
```

Scenario: Register New User

- A new user tries to register with our web application with a username
- Instead of waiting for the user to complete the entire Form, Ajax can be used to check if the chosen username is allowed in our system
- The user can be informed immediately if the chosen username is not allowed



Detect Lost of Focus

First, we add an event-listener on the "username" textbox to detect him leaving the input field

```
window.onload = function()
{
    let elem = document.getElementById("username");

    if (elem)
    {
        elem.addEventListener("blur", function() {
            checkIsUsernameOkay(elem.value);
        });
    }
}
```

Adding an Event Listener to detect when focus has left the “username” textbox

Making a AJAX Request

Then, use JavaScript to create and send a AJAX request to our Server to check if the entered username can be used

```
function checkIsUsernameOkay(username)
{
    let xhr = new XMLHttpRequest();

    xhr.open("POST", "/Register/IsUsernameOkay");

    xhr.setRequestHeader("Content-Type",
                         "application/json; charset=utf8");

    xhr.onreadystatechange = function() {
        ...
    }

    let data = {
        "Username": username
    };

    xhr.send(JSON.stringify(data))
}
```

Request AJAX capability from the JavaScript Engine

The Controller and Action Method to receive the data

onreadystatechange() is called when there are changes to the state of our request

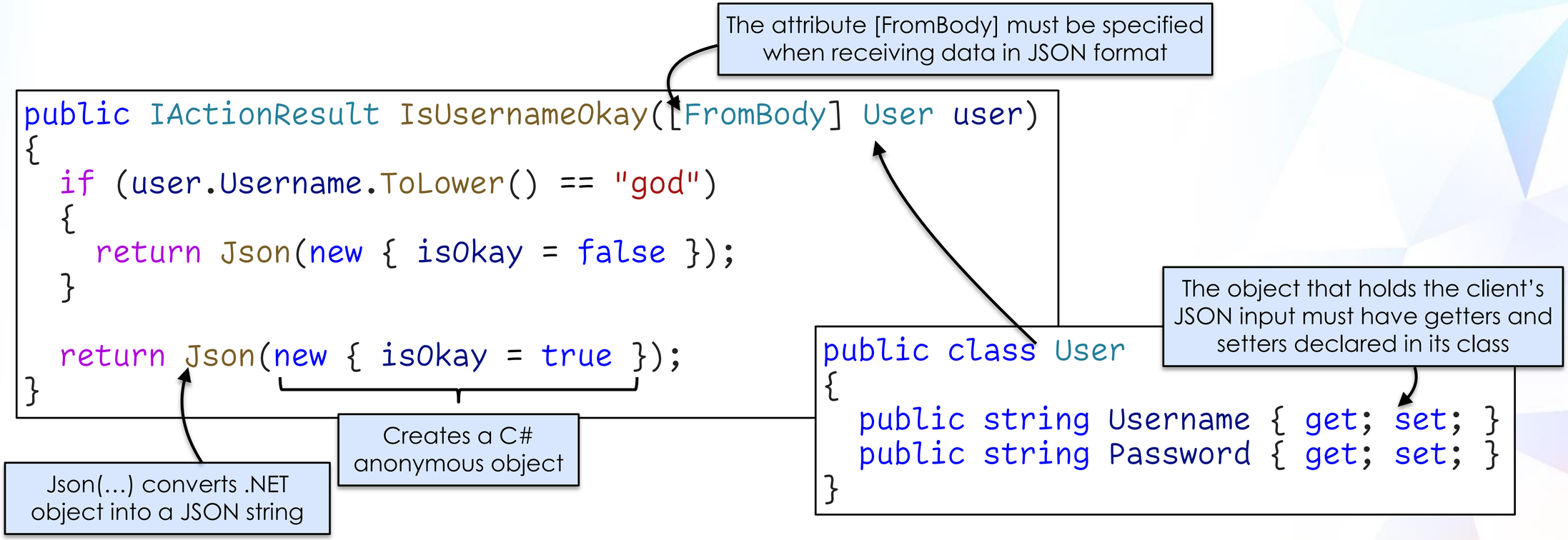
Tell the server that our data is in JSON format

Packing data into a JavaScript object

JSON.stringify(...) to convert a JavaScript object into a JSON string

Server receives a AJAX request

Our .NET web application accepts the AJAX request and binds the incoming data against an object's properties



Handling the Response for an AJAX request

Finally, we handle the Server's response to our AJAX request with JavaScript

```
function checkIsUsernameOkay(username) {
    ...

    xhr.onreadystatechange = function() {
        if (this.readyState == XMLHttpRequest.DONE) {
            // receive response from server
            if (this.status != 200) {
                return;
            }

            // convert from JSON string to JavaScript object
            let data = JSON.parse(this.responseText);

            // check availability response
            if (data.isOkay == false) {
                alert("Username is not allowed.");
            }
        }
    }
    ...
```

XMLHttpRequest.DONE means a complete response has been received

'this' points to our XMLHttpRequest object

HTTP Status of 200 is a response from the web server that our web request has been handled successfully

To convert the JSON string into a JavaScript object

AJAX with name-value pairs

AJAX can also be sent as name-value pairs, with .NET binding the incoming data to matching parameter names of the target Controller/Action Method

Client (JavaScript)

```
function checkIsUsernameOkay(username) {
    let xhr = new XMLHttpRequest();

    xhr.open("POST", "/Register/IsUsernameOkay");
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

    xhr.onreadystatechange = function() {
      ...
    }


    xhr.send("username=" + username + "&suggest=true")
}
```

Specify that our data will be encoded in name-value pairs format

```
public IActionResult IsUsernameOkay(string username, bool suggest ) {
    ...
}
```

Server (.NET/C#)

THE END