前端網絡開發人員課程
（二）進階網絡程式設計

12. Create Your Own Website

Presented by Krystal Institute

Learning Objective

- Understand how to navigate through different html pages
- Create a fully functional website

Content

12.1 Navigate through pages

Navigation

- Before, we talked about using the action property of a form element or the href property of the <a> element to access other pages, but we haven't tried it before

- You will learn how to navigate through html files and pages as well as sending form data with only frontend JS in this lesson, and these will be very useful for the assignment to come

Navigation: href Activity

- To navigate, first we need to create 2 html files, in this example, we will name it L15 A.html and L15 B.html

```html
<body>
<h1>A</h1>
</body>
```

- Firstly, add a <h1> element with the name with the respective letter A or B on the page to make it easy to distinguish between pages

```html
<body>
<h1>B</h1>
</body>
```

Navigation: href Activity

- Create an <a> element with any text inside, and add the href property in the A file

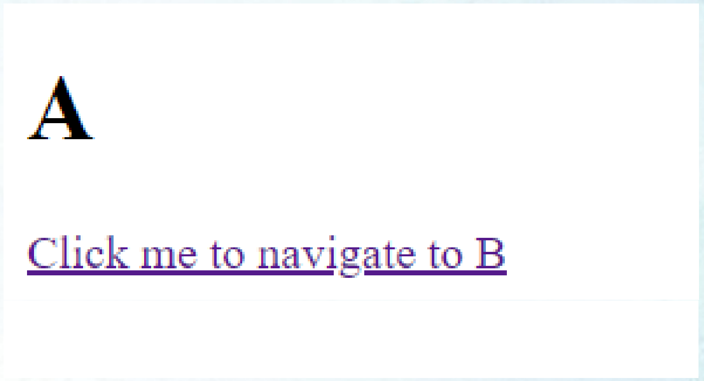
- In the href property, type in the name of the other html file, in this example. We will navigate from A to B, so type in L15 B.html

```
<body>
<h1>A</h1>
<a href="L15 B.html">Click me to navigate to B</a>
</body>
```

Navigation: href Activity

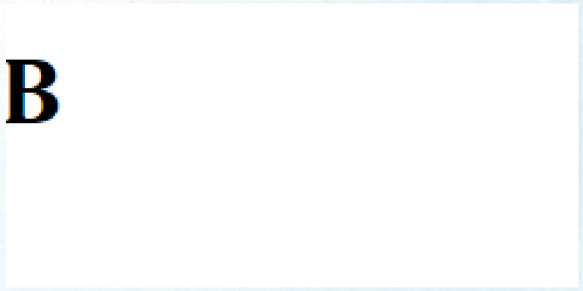
- Now upon clicking the link, you will be redirected to B's webpage!

- Make sure both websites are inside the same file



A

Click me to navigate to B



B

Navigation: form Activity

- To navigate with form data, using the get method will put form data on the url link, we can utilize this to move data between html pages

```
<body>
<h1>C</h1>
</body>
```

- To start, create 2 new html files called L15 C.html and L15 D.html
- We will be transferring data from C to D

```
<body>
<h1>D</h1>
</body>
```

Navigation: form Activity

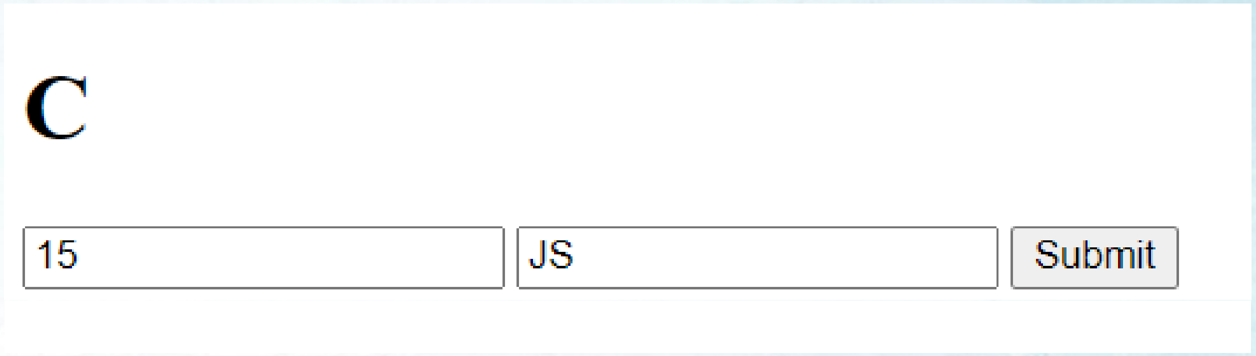
- Create a form with 2 inputs and a submit button on the C file

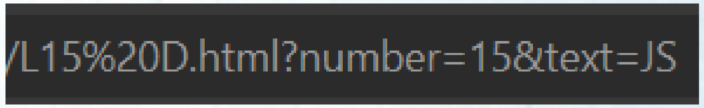
- In the method property of the form, use "get" since we want to have the data passed onto the url
- In the action method, put the name of the D file

```
<h1>C</h1>
<form action="L15 D.html" method="get">
    <input type="number" name="number">
    <input type="text" name="text">
    <button type="submit">Submit</button>
</form>
```

Navigation: form Activity

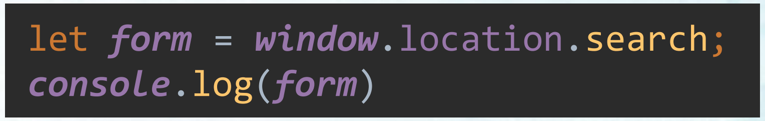
- When you press the submit button, you will see on your url the name and value of every input inside your form
- To extract that, we need to add the <script> tag on file D




```
/L15%20D.html?number=15&text=JS
```

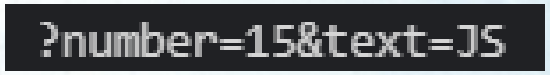
Navigation: form Activity

- Window.location is a property that represents the current url of the page, using it will return the whole url page



```
let form = window.location.search;
console.log(form)
```

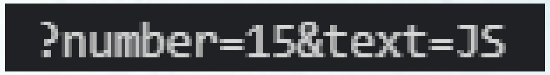
- Window.location.search only returns the get data



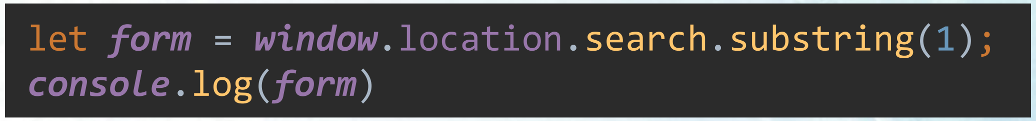
```
?number=15&text=JS
```

Navigation: form Activity

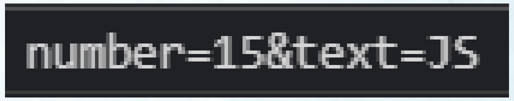
- In the url, the ? Separates the link and the query data, which is the form data we need, to get rid of ?, use the substring method on the search property, the number argument determines how many characters it will ignore, use 1 to skip the ?
- The form data is passed as a string, so we need to parse it



```
?number=15&text=JS
```



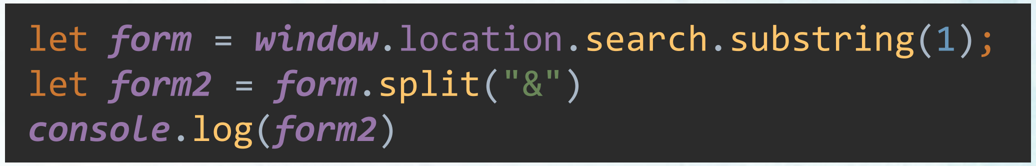
```
let form = window.location.search.substring(1);
console.log(form)
```



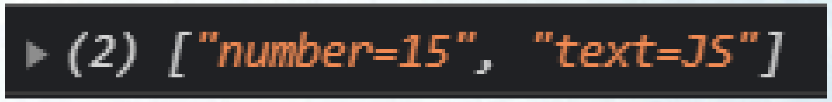
```
number=15&text=JS
```

Navigation: form Activity

- To separate each input, use the split() method along with a delimiter, which is &
- The method will split the string by the selected delimiter, and put all of it as elements in an array



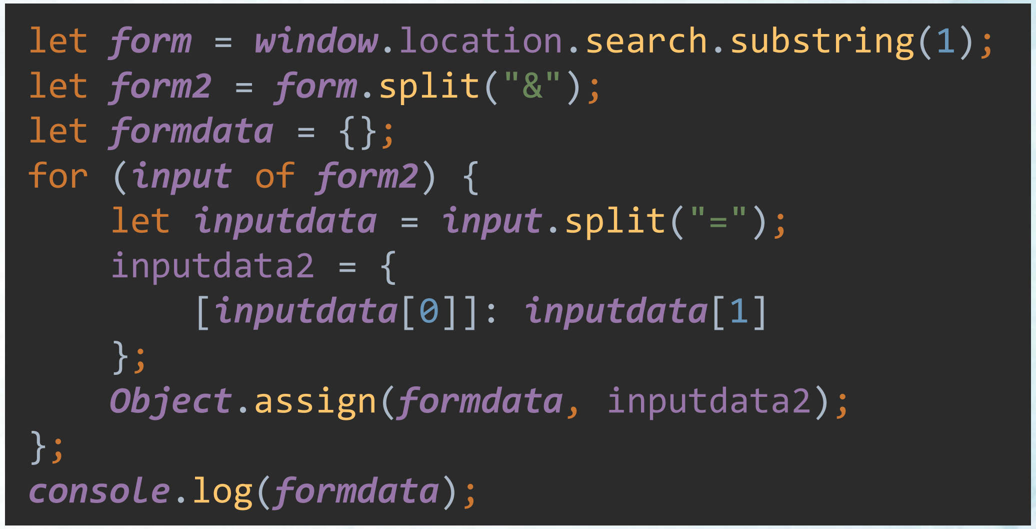
```
let form = window.location.search.substring(1);
let form2 = form.split("&")
console.log(form2)
```



```
▶ (2) ["number=15", "text=JS"]
```

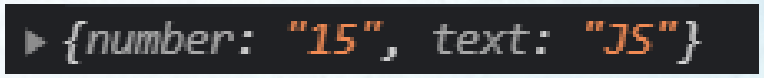
Navigation: form Activity

- To further split the name and value of each input, use a for loop and separate each input inside the loop with split()
- To make it even tidier, add them into an form object



```
let form = window.location.search.substring(1);
let form2 = form.split("&");
let formdata = {};
for (input of form2) {
    let inputdata = input.split("=");
    inputdata2 = {
        [inputdata[0]]: inputdata[1]
    };
    Object.assign(formdata, inputdata2);
};
console.log(formdata);
```

- Do not send personal information this way!



```
▶ {number: "15", text: "JS"}
```

Navigation: href

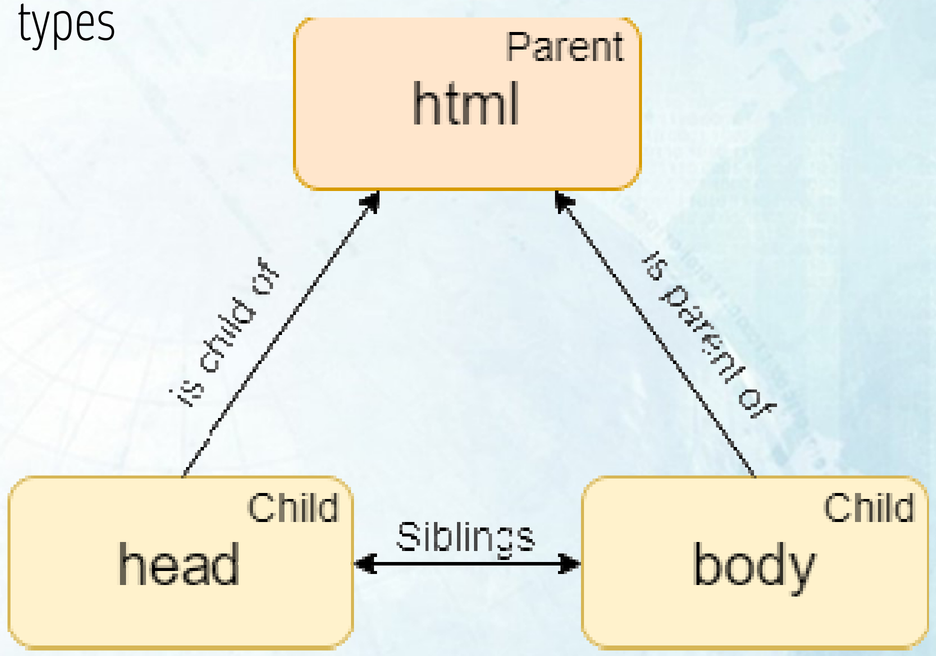
- It is also possible to redirect to another webpage within the script of the webpage
- Window.location.href shows the url of the webpage, assign a new html page to it to redirect the page
- Form data can also be added manually this way

```
window.location.href = "www.xxx.xxx...";
// remember to use the full url
```

12.2 Revise on previous lessons

DOM: Nodes and Elements

- Nodes are an abstract concept that are split into 3 main types
- Text nodes: includes texts, whitespace and newline
- Element nodes: includes elements
- Comment nodes: includes comments
- The relationship between each node are the same as a traditional family tree



Selecting Elements

- getElementbyId finds element by their id
- getElementsbyName returns a collection of elements by their name
- getElementsByTagname returns a collection of elements by their tag
- getElementsByClassName returns a collection of elements by their class
- querySelector / querySelectorAll returns an element / a collection of element by their CSS selector

Traversing Elements

- parentNode returns the parent of a specified node
- firstChild / firstElementChild returns the first child / element child of the parent node
- lastChild / lastElementChild returns the last child / element child of the parent node
- childNodes / children returns a collection of all child nodes / child element nodes of the parent node
- nextElementSibling returns the next sibling in the list of elements
- previousElementSibling returns the previous sibling in the list of elements

Manipulating Elements

- createElement(Tag) returns a new element without the specified element type
- appendChild(parentElement) moves a node onto the end of the list of nodes in the parent node
- element.textContent gets / sets the text node of an element
- innerText gets / sets human-readable text only
- innerHTML gets / sets the HTML markup of a specified element

Manipulating Elements

- DocumentFragment creates a lightweight version of the document for easy modification and appending
- insertBefore inserts a new node before the specified child node
- append() inserts a set of nodes after the last child of the specified parent node
- prepend() inserts a set of nodes before the first child of the specified parent node
- insertAdjacentHTML inserts texts adjacent to the specified element

Manipulating Elements

- replaceChild replaces the old node with a new node
- cloneNode clones an element and returns it
- removeChild removes a child node from a parent node

Attributes and Properties

- Attributes and properties define a element
- element.attributes returns a collection of attributes
- Data-* attributes are reserved for developer use
- setAttribute / getAttribute sets / gets the value of an attribute
- removeAttribute removes an attribute from a specified element
- hasAttribute returns a Boolean that determines if an element has a specified attribute

Styling

- element.style changes the style property of an element
- cssText and setAttribute can be used to set multiple styles at once
- getComputedStyle retuns the computed style of an element
- className returns a space-separated string of CSS classes
- classList retuns a collection of CSS classes

Events

- An event listener "listens" to user actions and triggers
- onclick attribute, onclick property, and addEventListener applies a event to an element
- removeEventListener removes an existing event listener
- event.target returns the element that triggered the event
- DOMContentLoaded / load triggers after parts / all of the browser has been loaded
- beforeunload / unload triggers before / after everything is unloaded

Events

- Mouse events triggers when you use your mouse
- mousedown / mouseup triggers when you press / release the mouse button
- click triggers after one mouseup and one mousedown
- Keyboard events triggers when you use your keyboard
- keydown / keyup triggers when you press / release the key
- Keypress triggers constantly when you hold character keys

Events

- event.key / event.code returns the character / character key that's been pressed
- Scroll events triggers when you use the scroll wheel or scroll bar
- scrollTop / scrollLeft returns the offset of the scrolling
- scrollIntoView scrolls elements into view
- focus / blur triggers when an element receives / loses focus

Events

- Multiples of the same event can be delegated by adding it onto the parent node
- dispatchEvent triggers the event from the code
- event.isTrusted returns if the event was triggered by user actions
- MutationObservers observes changes to an element

ES6 Syntax

- ES6 is a programming language standard for JavaScript, making it more modern and readable
- The let word declares a new variable like var, but it is block-scoped
- A block-scoped variable works well in a asynchronous environment
- Variables declared with the let keyword cannot be referenced outside of its block
- Let doesn’t allow you to redeclare a variable
- Let cannot hoist and be referenced before declaration

ES6 Syntax

- The const keyword works similar to the let keyword, except the variable created is read-only
- You cannot reassign a constant, however you can reassign properties inside a constant as it doesn't change the constant itself
- Default Parameters allows for parameters of a function to default to a certain value if no value or undefined is passed on that parameter

ES6 Syntax

- The rest parameter has a prefix of … and it represents any number of arguments after the rest syntax as an array
- The rest parameter can only be used as the last parameter in a function
- The spread operator has the same prefix … and it is used to unpack elements inside an array
- The spread operator can be used anywhere inside an array

ES6 Syntax

- If the property name and value of an object is the same, only one has to be used to create an object
- Using square brackets at the property name of an object treats everything inside the brackets as a string, in case of an variable, the value of the variable will be used and treated as string
- The property name of an function inside an object doesn't have to be included in ES6 if the name of the property is the same as the function name

ES6 Syntax

- For…of is a new way of iterating over iterable objects, it works very similar to a for loop, but with much less coding
- Array.entries can be used with for…of to get the index and the value of the element in the array
- Octal Literals have a prefix 0o (zero followed by the letter o) followed by octal numbers
- Binary Literals have a prefix 0b followed by binary numbers
- Template Literals are defined by backticks, it allows for insertion of variables inside a string, as well as allowing single and double quotes inside one

Array Destructuring

- Destructuring assignment allows for assigning multiple variables at the same time using arrays
- The rest syntax can be used on array destructuring, where it takes all the remaining values as an array and assigning it to a variable
- Default parameters can be used to assign a default value to a variable if array destructuring contains a undefined value
- Nested array destructuring is also possible if the variables have the exact same format as the values

Arrow Functions

- Arrow functions has a prefix of an arrow => and it speeds up the creation of functions
- The brackets containing parameters should be used even without parameters before the arrow prefix unless the parameter is the only variable used inside the function
- If the block syntax is used after the arrow prefix, the return keyword will have to be used
- Line breaks are not allowed in arrow functions unless the arrow prefix is placed before the line break
- Modules are extensions of the JS script, that allows you to import and export other methods, variables for use

Assignment

- Create a simple online shop, in any formats you like
- It will consist of a few elements:
- A homepage
- A products and product details page
- A login page

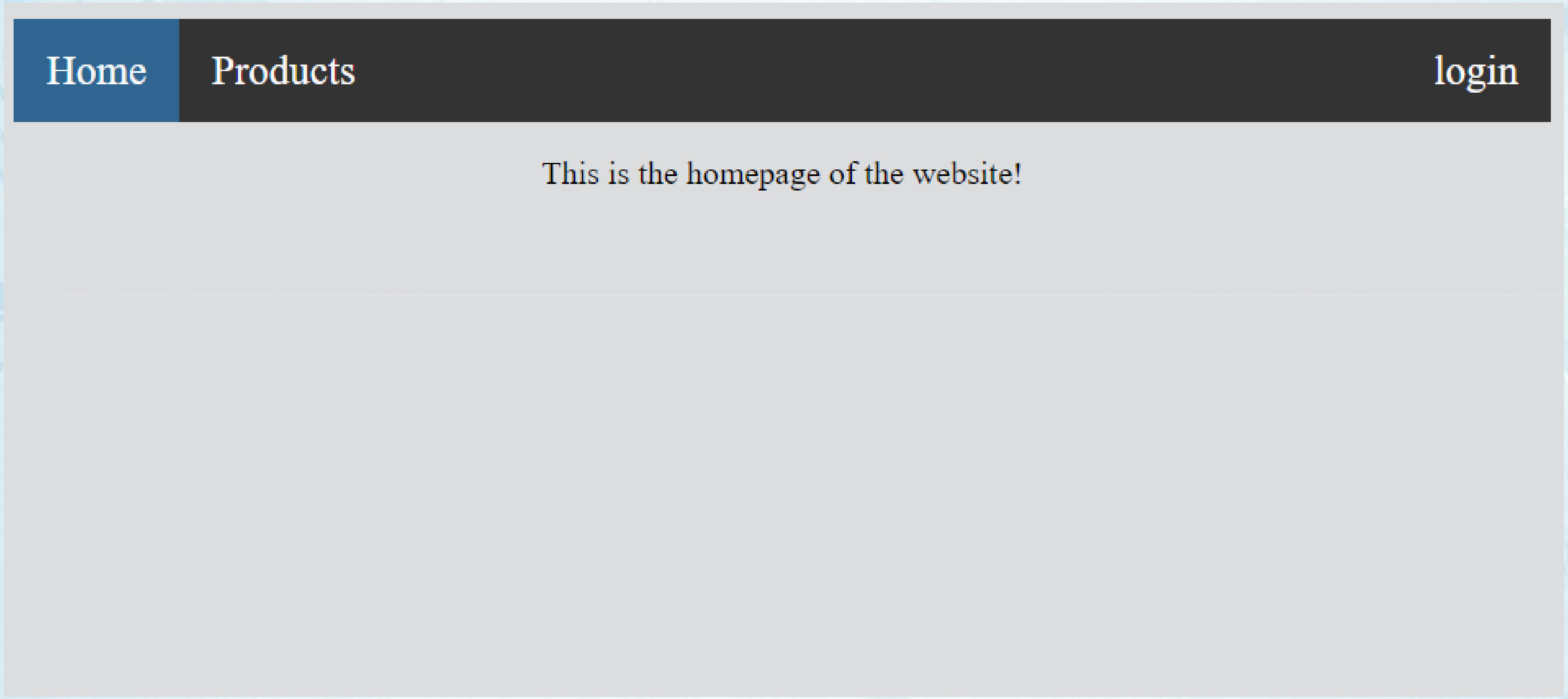
- Don’t hesitate to search for answers online or ask for help!

Assignment Details

- Home page: a home page is needed for every website, make it appealing to attract customers!
- Provide a navigation bar that can go to the home page, the products page and the login page
- Registering an account will require backend JS to work so only include the login button for now

Front-end Web Developer

Assignment Details



This is the homepage of the website!

Assignment Details

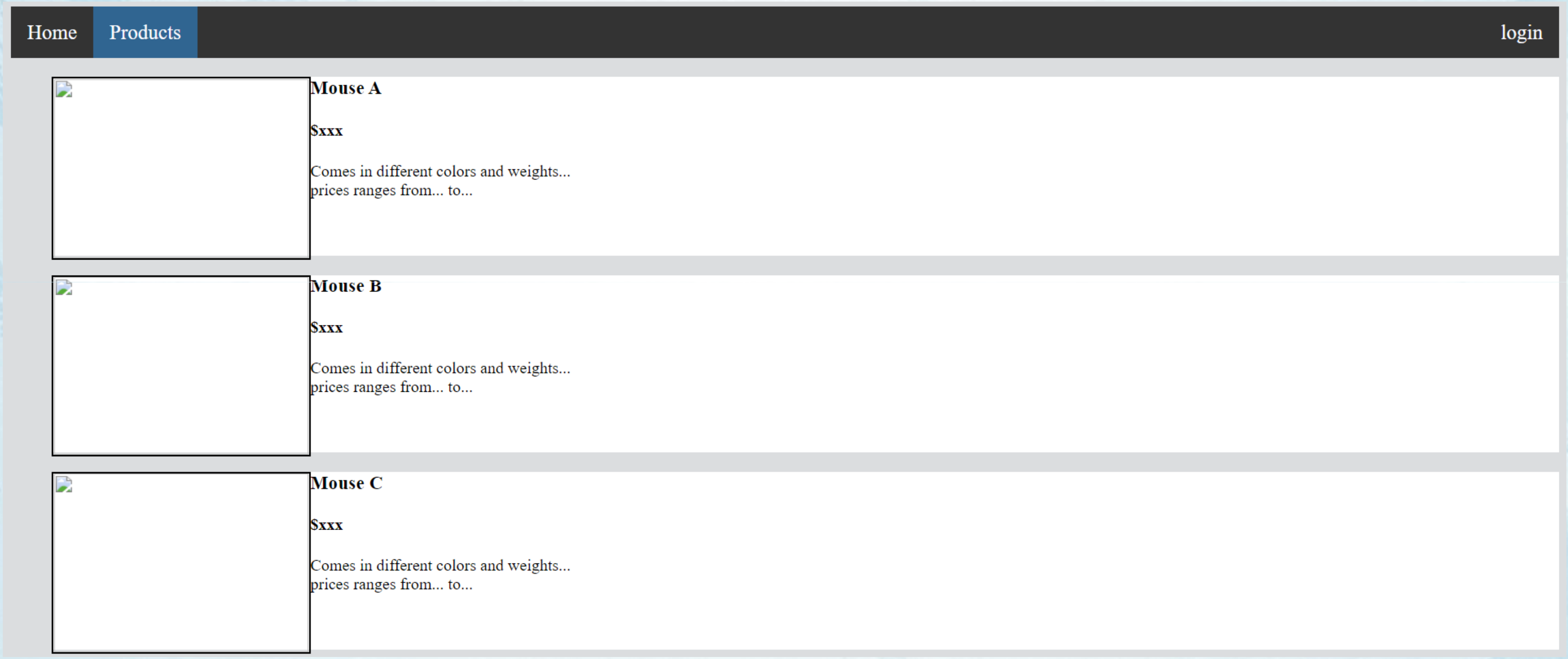
- Products page: keep the navigation bar throughout the whole website
- The page should contain a list of products (fill it in with anything random) and its pictures, since you decide what images to provide, you can either search for images online or set it to no images for now
- Use a black border to mark out where the image should be

Assignment Details

- it should show the name of the product, the pricing, and description
- The product listing should be clickable, and will direct into a specific product page (hint: manually add form data to specify the product page)

Front-end Web Developer

Assignment Details



Mouse A

$xxx

Comes in different colors and weights...
prices ranges from... to...

Mouse B

$xxx

Comes in different colors and weights...
prices ranges from... to...

Mouse C

$xxx

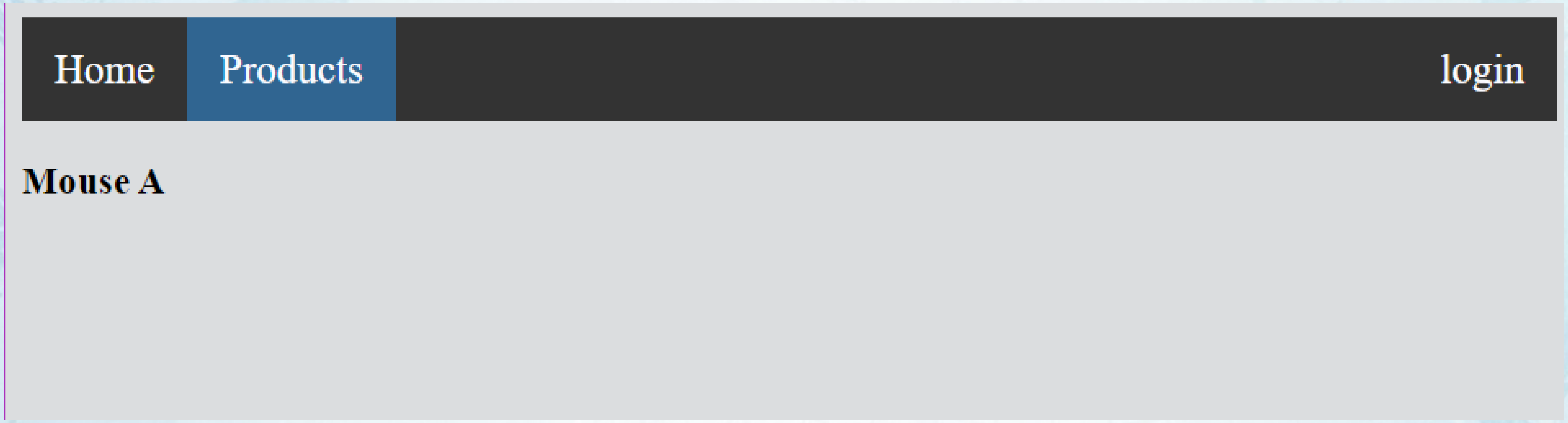
Comes in different colors and weights...
prices ranges from... to...

Assignment Details

- Specified products page: normally, the details would be sent from the database
- In this case, give some simple indication to show which product the user has clicked on
- This is to show that the redirecting works
- Hint: use window.location.search to get the specified product the user clicked on

Front-end Web Developer

Assignment Details

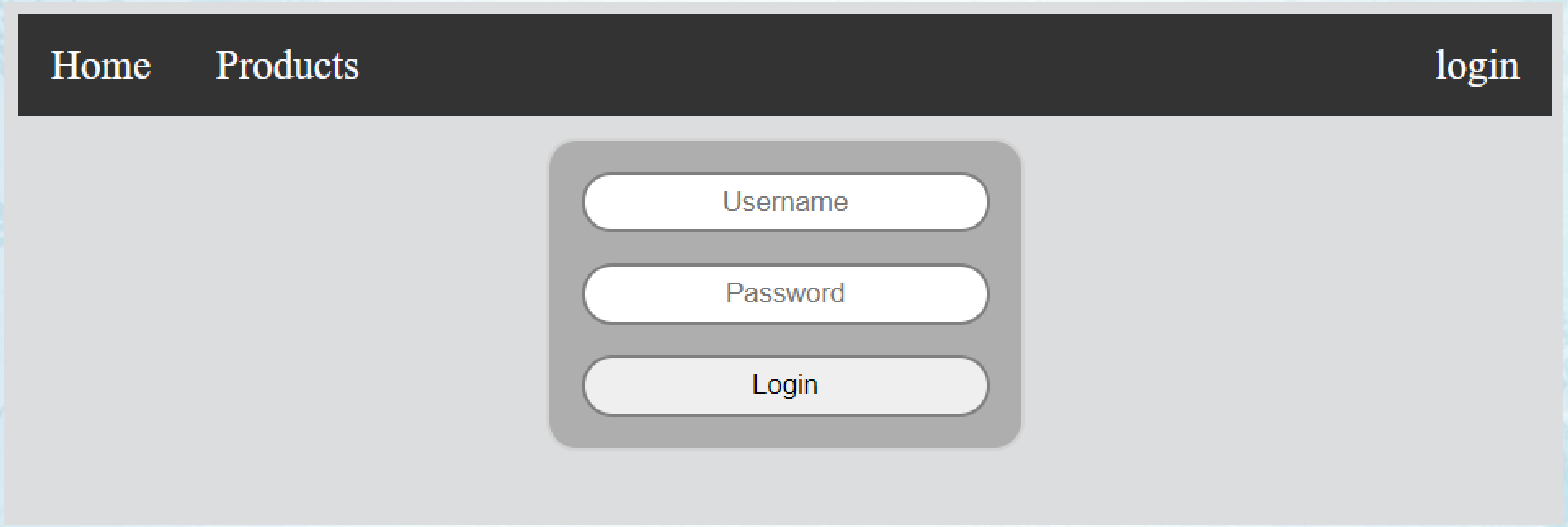


Mouse A

Assignment Details

- Login page: create a standard login page (freely style your buttons and inputs!)
- It should contain a username and a password input, as well as the login button
- The webpage should also check for invalid accounts, and handle incorrect logins
- In this assignment, have the correct username and the password set as…
    - Username: user
    - Password: 1234
- Redirect it to the homepage when the login is correct

Assignment Details



References

- Use these if you need more explanations!
- https://www.javascripttutorial.net/es6/
- https://javascript.info/

- Use this if you need more specific answers!
- https://stackoverflow.com/