# HSBC Technology Graduate Training

## Web Programming

Day 5

Friday 30 October 2020

# Contents

# EVENT LISTENERS & EVENT HANDLERS

- Similar to how we can create objects in Java, we create objects in HTML.
- We create objects in HTML with tags.
- The difference is, objects in Java are created in RAM so we can't see it.
- Objects in HTML are visible, for instance, a button etc.

```
1 Name: <input type="text" name="name">
```

Name:

# EVENT LISTENERS & EVENT HANDLERS

- *<input>* is an example of a tag in HTML that creates an input object.
- Such objects have some built-in functionality which we cannot change.
  - For example, the input object has an <u>event listener</u> for <u>onclick</u> events.
  - Such functionality can be found in the documentation of HTML.

# EVENT LISTENERS & EVENT HANDLERS

- **What are event listeners?**
  - **An event listener emits an event when some pre-defined action occurs.**
  - **It *listens* for an event to occur.**
  - **For instance, the onclick event listener emits an event when someone clicks on the input object.**
- **We can use event emitters to perform some actions that we can define.**
- **We do this using an event handler.**

# EVENT LISTENERS & EVENT HANDLERS

- We can define some code that will execute upon an event with <u>JavaScript.</u>
- JavaScript has NOTHING to do with Java.
- JavaScript is a web-scripting language.
- The example below shows how we can send an alert to the browser when someone clicks on the *name* input.

```html
<html>

    <script>
        function openAlert() {
            alert("Hello world! You clicked the input");
        }
    </script>

    Name: <input type="text" name="name" onclick="openAlert()">

</html>
```

Javascript code must live in a .js file or in-between the script tags in the .html file.

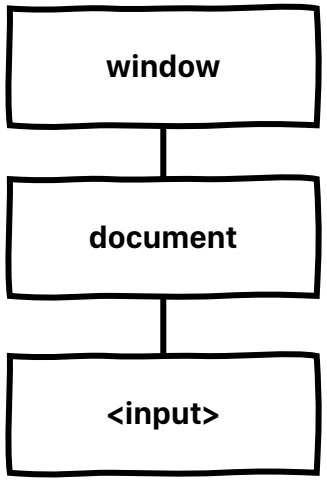
Here, we are telling the input object to call the function **openAlert()** when the onclick event listener emits an event (i.e. when someone clicks on the input field).

# DOM

# DOM

- A DOM (Document Object Model) is a representation of a web page.
- It is a tree of all objects on a webpage.
- The root of the tree starts with a window object.
- The document object is a child of the window object.
- The HTML objects that we create are children of a document object.

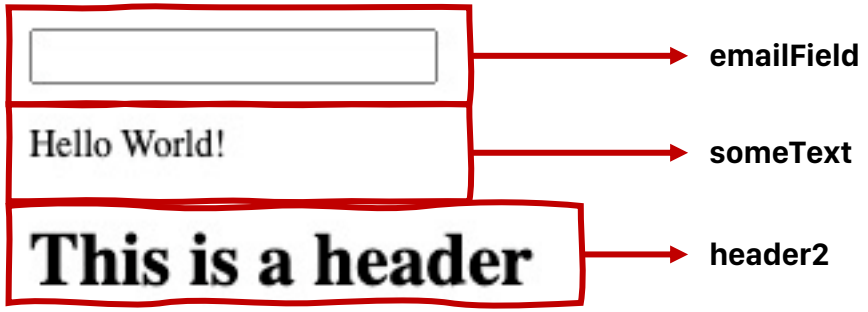
```
1 Name: <input type="text" name="name">
```



# DOM

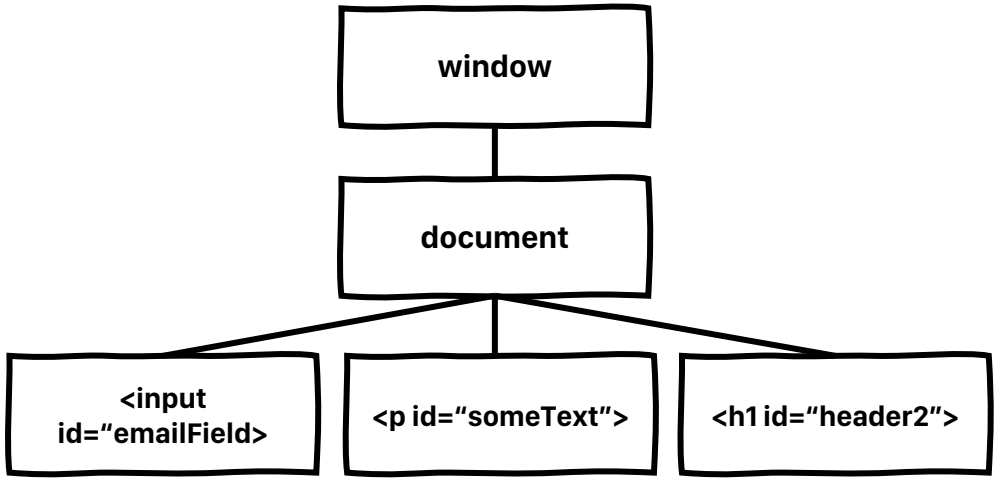
- We can provide an <u>identifier</u> or <u>id</u> to an object on the DOM so we can manipulate it with JavaScript.
- To assign an <u>identifier</u> to an object, we must set the attribute <u>id</u>.
- The example below shows us how we can assign <u>identifiers</u> to various HTML objects.

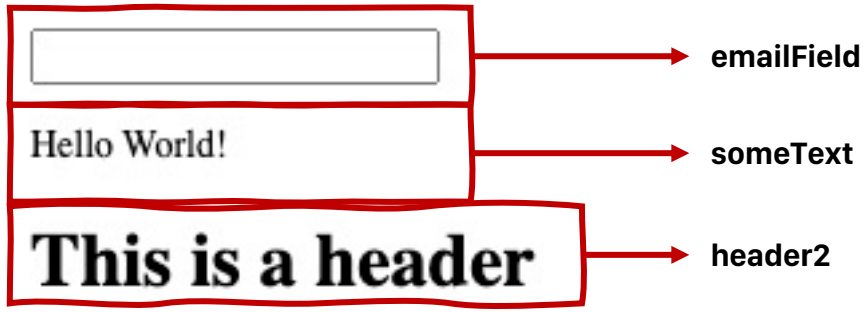
```html
<html>
    <input type="text" name="email" id="emailField">
    <p id="someText">Hello World!</p>
    <h1 id="header2">This is a header</h1>
</html>
```



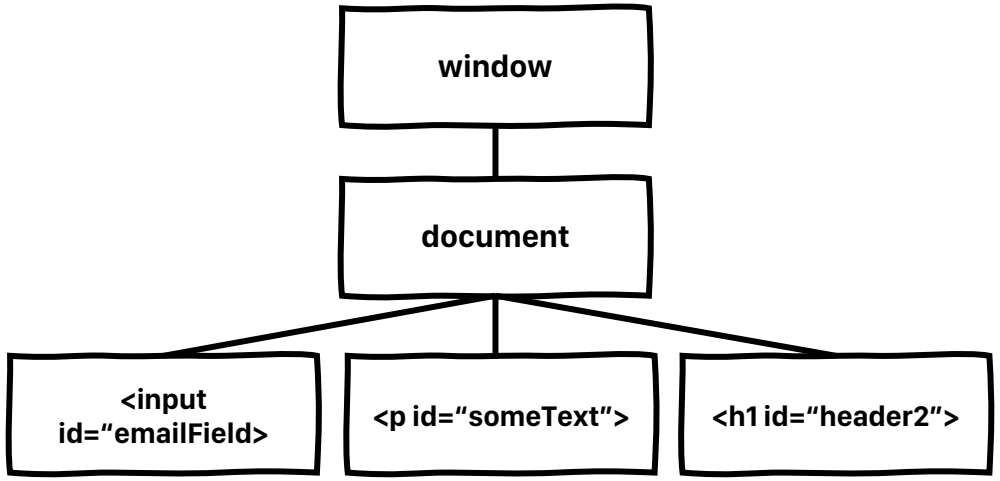
# DOM

- We can use some built-in JavaScript functions that will help us retrieve an (reference to) object with <u>identifiers</u>.
- Below is a visual representation of the DOM for the previous example.



- To get a reference to the `emailField` object we can use the following Javascript code.




```
1 let emailFieldRef = window.document.getElementById("emailField");
```

- We do not need to declare the type of a variable in JavaScript, so we use let to create a variable.
- **document** provides a function **getElementById(x)** that will allow us to retrieve the reference to an object with an id **x**.
- Once we have a reference to an object, in this case, an input object, we can perform actions upon it, such as changing the value of the field.

# DOM

- We can harness the ability to retrieve DOM objects with Javascript and programmatically perform some action upon some event.
- The example below shows some HTML and Javascript that will change the value of the input field emailField to "Hello World" when it is clicked.

```html
<html>
    <script>
        function changeValueOfInputField() {

            // Get reference to input object with ID emailField.
            let emailFieldRef = window.document.getElementById("emailField");

            // Sets value of input field to "Hello world"
            emailFieldRef.value = "Hello world";

        }
    </script>

    Email: <input type="text" name="email" id="emailField" onclick="changeValueOfInputField()">

</html>
```

#  vs. 

# DIV VS. SPAN

- The  tag creates an object in the DOM.
- Acts as a ‘container’ for other objects.
- All objects between the  tags will be children of that div.

```
<div>Hello World</div>
<p>Hello world</p>
```

Hello World

Hello world

# DIV VS. SPAN

- The  tag acts in the same way.
- The difference is that the  is an *inline* element as opposed to a *block* element.
- A  element will not move the next element to a new line.

```
<span>Hello world</span>
<p>Hello world</p>
```

Hello worldHello world

# Styles

# STYLES

- We can set the style (look) of objects with the style attribute.
- Some styles we can set:
  - Positioning with top, bottom, left, right.
  - Background colour with background-color
  - Font with font-family, font-size, font-weight, font-style

```
<div style="background-color: red">
    Hello world!
</div>
```

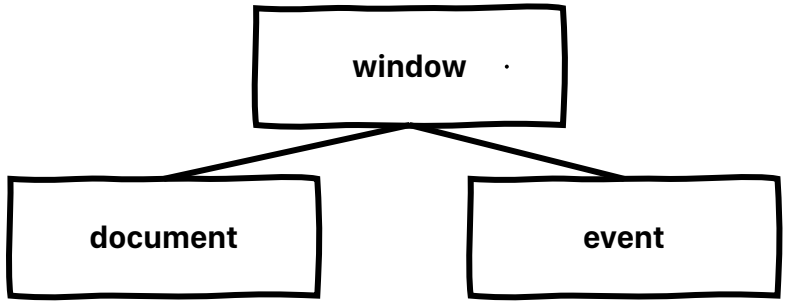
- We can set the absolute position of an element with the position(s) attributes.
- We can set the absolute position of an element by first setting the position of an element to ‘absolute’.
- Then, we can use the top, bottom, left and right attributes to manipulate the position of an element relative to its parent.
- Top: 0 means top of the page. Top: 50 means 50 away from the top of the page.

```
<div style="position: absolute; top: 30">
    Hello world!
</div>
```

# Events

# EVENTS

- In addition to the <u>document</u> object, there is also a <u>event</u> object.
- The <u>event</u> object contains information about the events that have been emitted in the <u>window</u>.
- For example, the <u>event</u> object contain information about the x and y positions of the cursor. We can access this property using window.event.clientX



- We can use the coordinate string for various means.
- The example below shows a simple script which prints the coordinate when an element is clicked.

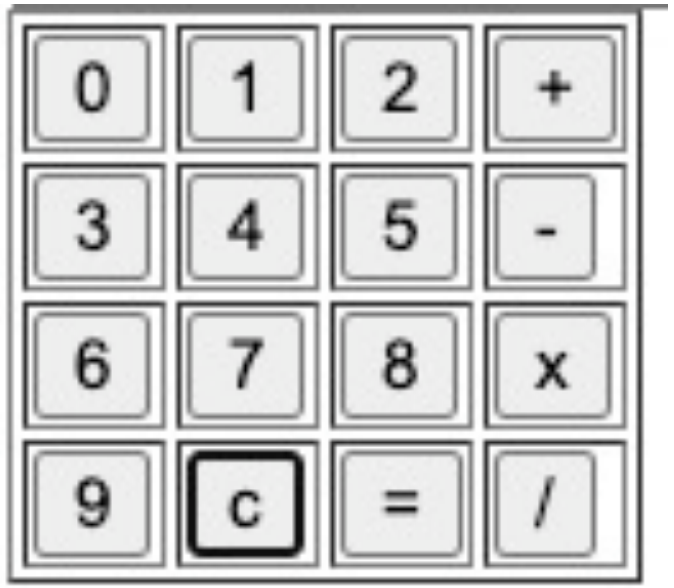
```html
<html>
<script>
    function displayCursorCoordinate() {

        // Create coordinate string.
        let coordinate = window.event.clientX + ', ' + window.event.clientY;

        // Print coordinate
        console.log(coordinate);

    }
</script>

<p onclick="displayCursorCoordinate()">

</html>
```

# Tables

# SUBSTRING

- We can create a table using HTML with the <table>, <td> and <tr> tags.

```html
<table border="1">
        <tr>
            <td><input type="button" value="0" onclick="addDigit(event)"></td>
            <td><input type="button" value="1" onclick="addDigit(event)"></td>
            <td><input type="button" value="2" onclick="addDigit(event)"></td>
            <td><input type="button" value="+" onclick="addOp('+')"></td>
        </tr>
            <td><input type="button" value="3" onclick="addDigit(event)"></td>
            <td><input type="button" value="4" onclick="addDigit(event)"></td>
            <td><input type="button" value="5" onclick="addDigit(event)"></td>
            <td><input type="button" value="-" onclick="addOp('-')"></td>
        <tr>
            <td><input type="button" value="6" onclick="addDigit(event)"></td>
            <td><input type="button" value="7" onclick="addDigit(event)"></td>
            <td><input type="button" value="8" onclick="addDigit(event)"></td>
            <td><input type="button" value="x" onclick="addOp('*')"></td>
        </tr>
        <tr>
            <td><input type="button" value="9" onclick="addDigit(event)"></td>
            <td><input type="button" value="c" onclick="clearScreen()"></td>
            <td><input type="button" value="=" onclick="equals()"></td>
            <td><input type="button" value="/" onclick="addOp('/')"></td>
        </tr>
    </table>
```



# Substring

# SUBSTRING

- JavaScript provides a method **substring(x, y)** that takes in two input parameters.
- The **substring()** method exists on a string.
- Below are some examples of **substring()**

```
<script>
  console.log("Hello World".substring(1, 4); // Prints ell
</script>
```

```
<script>
  let ref = "Hello World";
  console.log(ref.substring(3, 4); // Prints l
</script>
```

# SUBSTRING

- How does substring work?
- Each character in a string is given a position.
- Substring will return a portion of a string.
- For instance, with the string below, calling substring(4, 7) will return “o w”