JavaScript & DOM

Using HTML Script Tag

```
<script>
Script code here
</script>
```

Use to tell the browser the beginning and ending point of scripting language in HTML Doc.

```
<script type="text/javascript">
JavaScript code here
</script>
```

```
<script type="text/javascript" src="yourfile.js">
</script>
```

<script> location

- Any number of <script> are allowed in the HTML document
- <script> can be placed in <head> or <body> or both
- Trick: placing scripts at the bottom of <body> improves speed of page rendering

External JavaScript

- In the external JavaScript file cannot contain <script>
- Advantages:
  - HTML and JavaScript are separated physically
  - Easier to maintain
  - Proxy server or browser caches can store frequently used JS file - speed up page loading

Variables

- Declaring variables: *var,let* keyword
- Variables are case sensitive
- Avoid using reserved as variables name
- The variable values (or type) can include number, string, Boolean and null
- JavaScripte allows virtually any value to be assigned to any variable
- Special characters can be used in string type variables (ex. \t, \n, \\, \", \')

Variables

- Examples

```
var web;
var str="web technology";
var str1='web technology';
let x=120;
var code=true;
var t=null;

y=200.5;
```

var VS let

	var	let
Declaring variable	Y	Y
Declare many var in 1 statement (separate by comma)	Y	Y
Re-declare var.	Y	N
Block scope	N	Y
Use var. before it is declared	Y	N

Hoisting Behavior

- Hoisting is the behavior of moving all declarations to the top of the current scope
  - All variables in the scope can be used right from the start of the scope (before the declaration of variables)
  - Only variables declared with var keyword
- Keyword 'let' also has this behavior
  - But that var. cannot be used before its declaring point
  - Temporal dead zone

Functions

- Declaring function

```
function functionname()
{
    code
}
```

- Function names are case sensitive
- The function name must begin with a letter or underscore and cannot contain any space

Functions

- Functions can have one or more parameters

```
function func1(var1, var2)
{  document.write("var1="+var1+", var2="+var2); }
```

Nameless function

- Sometimes called anonymous function
- Function without name
  - Ex: (function() { ... });
- Usage:
  - Immediately invoked function
    - <button onclick="(function() { alert('Hello World'); }) ();">Click</button>
  - Using anonymous functions as arguments
  - Assign the function to var. for calling later
    - let test = function() { alert('Hello World'); }; test();
- Arrow function is a shorthand for declaring anonymous function
  - let test = () => alert('Hello World');

Operators

- Mathematical Operators
  - +, -, *, /, %, ++, --
- Assignment Operators
  - =, +=, -=, *=, /=, %=
- Comparison Operators
  - ==, !=, >, <, >=, <=, ===, !==
- Logical Operators
  - &&, ||, !, &, |, ^, >>, >>>, <<
  - >> preserved the sign bit while >>> doesn't

Conditional Statements

- if/else

```
If (condition) {
    javascript statement
}
else {
    javascript statement
}
```

Conditional Statements

- switch

```
switch(varname) {
    case "X":
            javascript statement;
            break;
    case "Y":
            javascript statement;
            break;
    default:
            javascript statement; }
```

Loops

- for
- while
- do ... while

Event Handlers

- Event is something that happens when viewer of the page perform some actions such as clicking a mouse button
- Event Handlers can be used to identify the occurring event and then perform a task or a set of task
- With Event Handler, the page can react to the action of the viewer

Event Handlers

- Each event handler responds to or applies to different objects (html elements)
- For example:

Event handler	Applies to:	Triggered when:
onAbort	Image	The loading of the image is cancelled.
onBlur	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, Window	The object in question loses focus (e.g. by clicking outside it or pressing the TAB key).
onChange	FileUpload, Select, Text, TextArea	The data in the form element is changed by the user.

Ref: https://www.elated.com/events-and-event-handlers/

Event Handler

- Using event handler in an HTML element

<input type="button" value="Click Me!" onclick="*JavaScript code here*" />

Example

```
<body>
<form>
<input type="button" value="Click Me!"
onclick="window.alert('Hi!');window.alert('Bye!');" />
</form>
</body>
```

Js_event_01.js

```
function hi_and_bye() {
window.alert('Hi!');
window.alert('Bye!');
}
```

```
<body>
<form>
<input type="button" value="Click Me!" onclick="hi_and_bye();"
/>
</form>
<script type="text/javascript" src="js_event_01.js"></script>
</body>
```

Js_event_01.js

```
function hi_and_bye() {
window.alert('Hi!');
window.alert('Bye!');
}
var hi_button = document.getElementById("say_hi");
hi_button.onclick = hi_and_bye;
```

```
<body>
<form>
<input type="button" value="Click Me!" id="say_hi" />
</form>
<script type="text/javascript" src="js_event_01.js"></script>
</body>
```

Event Handlers

- The blur event: onblur

Example

```
<form>
<input type="text" onblur="window.alert('Hey! Come back!');"
/><br />
<input type="text" />
</form>
```

Event Handlers

- The click event: onclick

Example

```
<body>
<form>
<input type="button" value="Do not Click Here"
onclick="window.alert('I told you not to click me!');">
</form>
</body>
```

Event Handlers

- The click event: onclick

Example

```
<body>
<a href="http://www.kmitl.ac.th"
onclick="return false;">Click me</a>
</body>
```

Event Handlers

- The focus event: onfocus

Example

```
<form>
Enter Your Name:
<input type="text" onfocus="window.alert('Don\'t forget to capitalize!');" />
</form>
```

Event Handlers

- The mouse over event: onmouseover

Example

```
<a href="http://www.kmitl.ac.th"
onmouseover="window.alert('mouse over');">
Try Clicking Me!</a>
```

Event Handlers

- The submit event: onsubmit

Example

```
<form onsubmit="window.alert('Thank You');">
What's your name?<br />
<input type="text" id="thename" /><br />
<input type="submit" value="Submit Form">
</form>
```

The Event object

- Automatically created when an event occurs
- A number of properties
  - Provide additional info about the event
  - For example:
    - Event.data
    - Event.height
    - Event.pageX/Event.pageY
    - Event.screen/Event.screen
    - Etc.

Ref: https://www.elated.com/events-and-event-handlers/

Web Workers

- A way to execute JavaScript in the background without affecting the performance
- Normally web workers are used for the CPU intensive script
- The script that run by a worker is always stored in a separated file
  - To avoid
    - Using global var
    - Directly access html element

To create web worker

```
if (typeof(w) == "undefined")
{
  w = new Worker("workers1.js");
}
```

To receive message from worker

```
w.onmessage = function(event){
    window.alert(event.data);
};
```

To terminate the worker

```
w.terminate();
w = undefined;
```

To send msg. out of the worker

```
postMessage(message);
```

Sending msg. into worker

- Main

…

```
const w = new Worker(“Worker1.js” );
w.postMessage(“Message”);
```

…

- Worker

…

```
self.onmessage = function(msg) {
console.log(“received: ”, msg);
}
```

…

Conclusion

- Data exchanges between main and workers done by onmessage event
- A worker can create sub-worker

Cookie

- Cookies are data stored in small text file
- Cookie were invented to help server remember info about the user
  - Ex: when user login, session info can be stored in a cookie
  - Cookies are saved in name-value pairs
  - When browser sends request to a server, cookies of that page of the server are added to request message

Cookie attributes

- There are many attributes
- Ex:
  - Expires: specifies expiry date of the cookie
  - Domain: specifies which host to be sent cookie to
  - Path: specifies which cookie to be sent to which URL
  - Etc.

Create cookie

- JavaScript can create a cookie
  - `document.cookie`
- Ex:

```
document.cookie = "user=Hello World";
document.cookie = "user=Hello World;
expires=Mon, 6 Feb 2023 12:00:00 UTC";
document.cookie = "username=John Doe;
expires=Mon, 18 Dec 2023 12:00:00 UTC;
path=/";
```

*assume that today is Sun, 5 Feb 2023

Read a Cookie

- Cookie can be read like this:

```
let i = document.cookie;
```

- The document.cookie will returen all cookie in one string ex:

```
cookie1=value1; cookie2=value2; cookie3=value3;
```

Change value of a Cookie

- Changing value of the cookie can be done in the same way as creating it

```
document.cookie = "user=Hello KMITL;
expires=Thu, 9 Feb 2023 12:00:00 UTC";
```

*assume that today is Sun, 5 Feb 2023

Delete a Cookie

- Cookie can be deleted by setting expires attribute to a past date

```
document.cookie = "user=Hello World;
expires=Fri, 14 Feb 2020 12:00:00 UTC";
```

Web Storage

- A way for web app. to store data locally
- Before HTML5, data are store in cookies
  - Cookies are included in every request
    - Less secure
    - Limited amount a data to be stored
- Web storage is per domain
  - All pages from the same domain can store and access the same data

Web Storage Objects

- There are 2 web storage objects
  - window.localStorage: stores data with no expiration
  - window.sessionStorage: stores data for one session (data deleted after the browser is closed)
- To check browser support

```
if (typeof(Storage) !== "undefined")
```

Storing data in Web Storage

- Data are stored in name/value pair
- To store
  - `localStorage.setItem("name", "John");`

  or
  - `localStorage.name="John";`

Retrieving data from web storage

- Ex:
  - `var n = localStorage.getItem("name");`

  or
  - `var n = localStorage.name;`

Removing data from web storage

- Item in web storage can be removed by
  - `localStorage.removeItem("name");`

sessionStorage Object

- sessionStorage Object can be used the same way as localStorage
- As mentioned earlier, sessionStorage stores data for only limited of time