CS1520 Recitation
Week 3
Javascript

http://cs.pitt.edu/~jlee/teaching/cs1520

Jeongmin Lee (jlee@cs.pitt.edu), Pranut Jain (prj17@pitt.edu)
Today

- Javascript
  - Function
  - Object
  - Events
  - Events Listener
- Quiz

Many examples are based on w3schools.com and tutorialspoint.com
Object
Example: a car object

- Let’s think about we have a car with these properties and methods.
  
  ○ Property
    - Name = Fiat
    - Model = 500
    - Weight = 1000 lbs
    - Color = white
  
  ○ What it can do
    - Start
    - Stop
    - Accelerate
    - Brake
Example: a car object

- And modeling these into JS properties and methods would be something like this ... (We will see how to define soon)

- Property
  - Name = Fiat
  - Model = 500
  - Weight = 1000 lbs
  - Color = white

- What it can do
  - Start
  - Stop
  - Accelerate
  - Brake

- Property
  - car.name = ‘Fiat’
  - car.model = 500
  - car.weight = 1000
  - car.color = ‘white’

- Methods
  - car.start()
Example: a car object

- First create a HTML page with paragraph element with id of ‘demo’.
- At this stage, nothing would be shown on webpage but h1 title.

```html
<!DOCTYPE html>
<html>
<body>
<h1> Car object demo page </h1>
<p id="demo"></p>
</body>
</html>
```
Example: a car object

- Let’s add javascript that create variable car.
- Add script block first, and then create a variable within there.

```html
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
  var car = "Fiat";
</script>
</body>
</html>
```
Example: a car object

- Now, we will make the p element to have name of car object ('Fiat').

```html
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
    var car = "Fiat";
    document.getElementById("demo").innerHTML = car;
</script>

</body>
</html>
```
Example: a car object

- Another way to make an object: with its properties.
- And let's see its type on html page.

```html
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
    var car = {type:"Fiat", model:"500", color:"white"};
    document.getElementById("demo").innerHTML = car.type;
</script>

</body>
</html>
```
Example: a car object

- Now, let’s work on **methods**!
- First, let’s have status and speed property.

```html
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
    var car = {type:"Fiat" model:"500", color:"white", status: "stop", speed: 0};
    document.getElementById("demo").innerHTML = car.status;
</script>
</body>
</html>
```
Example: a car object

- Let’s create a method of start. It changes status.

```html
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
    var car = {type:"Fiat", model:"500", color:"white",
    status: "stop", speed: 0, start(){this.status ="running"}};

    document.getElementById("demo").innerHTML = car.status;

    car.start();
    document.getElementById("demo").innerHTML = car.status;

</script>

</body>
</html>
```
Events
Event

- HTML events are "things" that happen to HTML elements.
- When JavaScript is used in HTML pages, JavaScript can "react" on these events.

- Common HTML Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onchange</td>
<td>An HTML element has been changed</td>
</tr>
<tr>
<td>onclick</td>
<td>The user clicks an HTML element</td>
</tr>
<tr>
<td>onmouseover</td>
<td>The user moves the mouse over an HTML element</td>
</tr>
<tr>
<td>onmouseout</td>
<td>The user moves the mouse away from an HTML element</td>
</tr>
<tr>
<td>onkeydown</td>
<td>The user pushes a keyboard key</td>
</tr>
<tr>
<td>onload</td>
<td>The browser has finished loading the page</td>
</tr>
</tbody>
</table>
Event

- Example

```html
<!DOCTYPE html>
<html>
<body>

<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {

    id.innerHTML = "Ooops!";

}

</script>

</body>
</html>
```
Event

- Example

```html
<!DOCTYPE html>
<html>
<body>
<h1 onclick="changeText(this)">Click on this text!</h1>
<script>
function changeText(id) {
    id.innerHTML = "Ooops!";
}
</script>
</body>
</html>
```

Event Handler function
The onload and onunload Events

- The **onload** and **onunload** events are triggered when the user enters or leaves the page.

- The **onload** event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

- The **onload** and **onunload** events can be used to deal with cookies.

Example:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onload
The onmouseover and onmouseout Events

- The `onmouseover` and `onmouseout` events can be used to trigger a function when the user mouses over, or out of, an HTML element:

```html
<!DOCTYPE html>
<html>
<body>
<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color:#D94A38; width:120px; height:20px; padding:40px;"/>
Mouse Over Me</div>

<script>
function mOver(obj) {
   obj.innerHTML = "Thank You"
}

function mOut(obj) {
   obj.innerHTML = "Mouse Over Me"
}
</script>
</body>
</html>
```

https://www.w3schools.com/js/tryit.asp?filename=tryjs_events_mouseover
Event Listener
EventListerner

- **EventListerner** is a function that attached to an **HTML element** and calls a **function** when **specified event** is triggered

```javascript
document.getElementById("myBtn").addEventListener("click", displayDate);
```
EventListener

- You can add many event handlers to one element.

- You can add many event handlers of the same type to one element, i.e two "click" events.

- You can add event listeners to any DOM object not only HTML elements. i.e the window object.
Syntax

element.addEventListener(event, function, useCapture);

- The first parameter is the **type of the event** (like "click" or "mousedown").

- The second parameter is **the function we want to call** when the event occurs.

- The third parameter is **a boolean value** specifying whether to use **event bubbling** or **event capturing**. This parameter is optional.
Examples

- Example 1: Click event attached to a button
  - [https://www.w3schools.com/js/tryit.asp?filename=tryjs_addEventListener_add](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addEventListener_add)

- Example 2: Attach many listener to an object
  - [https://www.w3schools.com/js/tryit.asp?filename=tryjs_addEventListener_add_many](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addEventListener_add_many)

- Example 3: Attach to window object (not HTML object)
  - [https://www.w3schools.com/js/tryit.asp?filename=tryjs_addEventListener_dom](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addEventListener_dom)

- Example 4: Passing a parameter
  - [https://www.w3schools.com/js/tryit.asp?filename=tryjs_addEventListener_parameters](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addEventListener_parameters)
Event Bubbling and Capturing

These are the way you order propagate multiple events in multiple HTML DOM.

E.g: `<p>` inside `<div>`
Event Bubbling and Capturing

These are the way you order propagate multiple events in multiple HTML DOM.

E.g: `<p>` inside `<div>` and attached two pop-up events on each box
Event Bubbling and Capturing

Event Bubbling
the inner most element's event is handled first and then the outer

the <p> element's click event is handled first, then the <div> element's click event.

https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_usecapture
Event Bubbling and Capturing

Event Capturing
the **outer most** element's event is handled **first** and then the inner

the `<div>` element's click event will be handled first, then the `<p>` element's click event.
Questions?