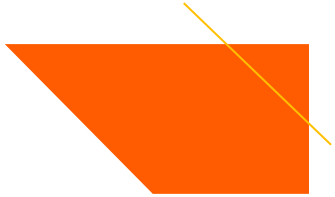
Web Application Development

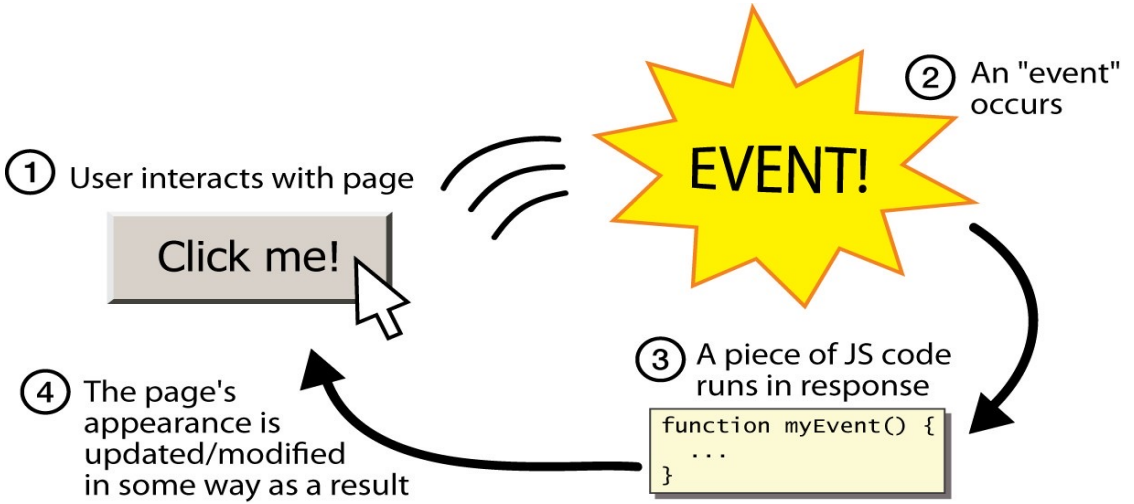
COSI 152A



Event-driven Programming

Event-driven programming

- JavaScript programs have no main. They respond to browser and user actions called events.
  - designed as a language to effectively respond to browser and DOM events
- Event-driven programming is writing programs driven by events



Responsive HTML controls

- To make a responsive UI control:
  1. choose the control (e.g. button) and event (e.g. mouse click) of interest
  2. write a JavaScript function to run when the event occurs
  3. attach the function to the event on the control

1.
```
<button>Click me!</button>
```

2.
```
function myFunction(){
   alert(”Button is clicked!");
}
```

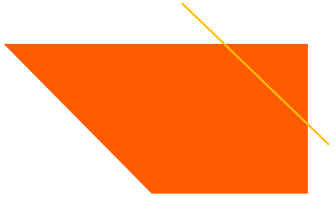
3.
```
<button onclick="myFunction();">Click me!</button>
<element attributes onclick="function();">...
```

Event handlers

- JavaScript functions can be set as event handlers
  - when you interact with the element, the function will execute
  - onclick is just one of many HTML event attributes we'll use

```
function myFunction(){
   alert("Button is clicked!");
}
```

- For more events visit here.
- Popping up an alert window is disruptive and annoying
  - a better user experience would be to have the message appear on the page...
  - but **HOW?**



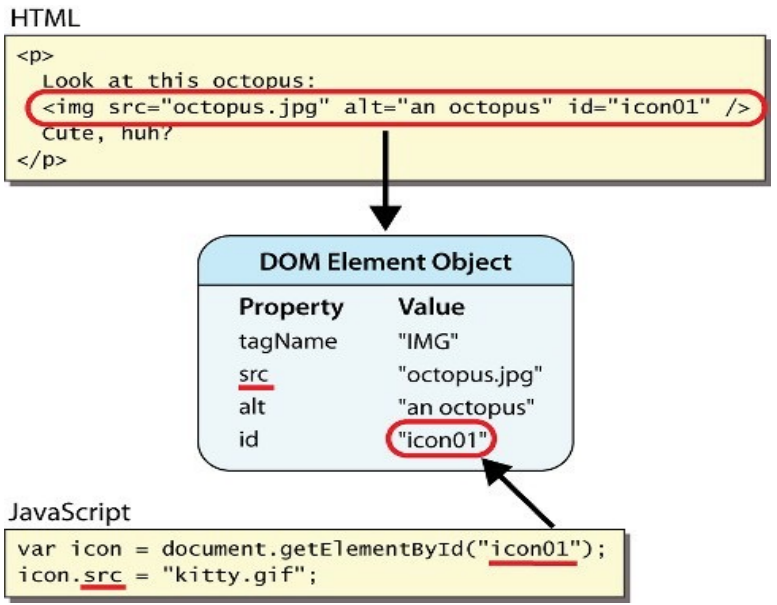
Document Object Model (DOM)

- The DOM is an API and JavaScript programmers use it to conveniently access and manipulate the HTML elements in code.
- We can examine elements' state
  - e.g., see whether a box is checked
- We can change state
  - e.g., insert some new text into a div
- We can change styles
  - e.g., make a paragraph red

DOM element objects

- Every element on the HTML page has a corresponding DOM object
- To access and modify the attributes of the DOM object:

```
objectName.attributeName = value;
```



Accessing elements

- `document.getElementById` returns the DOM object for an element with a given id.
- Browser automatically updates the screen when a DOM object is changed.
- The **value** property changes the text in most form controls.

```
const name = document.getElementById("someId");
```

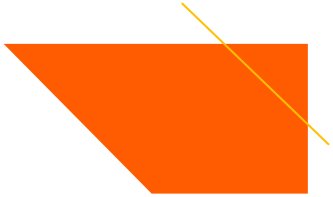
```
<button onclick="changeText();">Click me!</button>
<input id="output" type="text" value="replace me" />
```

```
function changeText(){
 const textbox = document.getElementById("output");
 textbox.value = "Hello, world!";
}
```

Accessing elements

- innerHTML property can change the text inside most elements.

```
<button onclick="swapText();">Click me!</button>
<span id="output2">Hello</span>
<input id="textbox2" type="text" value="Goodbye" />

function swapText() {
 var span = document.getElementById("output2");
 var textBox = document.getElementById("textbox2");
 var temp = span.innerHTML;
 span.innerHTML = textBox.value;
 textBox.value = temp;
}
```



DOM object properties

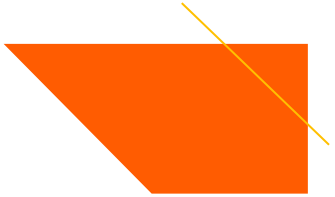
Property	Description	Example
tagName	element's HTML tag	mainDiv.tagName is "DIV"
className	CSS classes of element	mainDiv.className is "foo bar"
innerHTML	Content in element	mainDiv.innerHTML is "\n <p>Hello…
src	URL target of an image	icon.src is "greatwall.jpg"

```
<div id="main" class="foo bar">
  <p>Hello, <em>very</em> happy to see you!</p>
  <img id="icon" src="greatwall.jpg" alt="GREAT WALL" />
</div>

const mainDiv = document.getElementById("main");
const icon = document.getElementById("icon");
```

DOM properties for form control

Property	Description	Example
value	the text/value chosen by the user	sid.value could be "1234567"
checked	whether a box is checked	fresh.checked is true
disabled	whether a control is disabled (boolean)	fresh.disabled is false
readOnly	whether a text box is read-only	sid.readOnly is false

```
<input id="sid" type="text" size="7" maxlength="7" />
<input id="fresh" type="checkbox" checked="checked" />
Freshman?
const sid = document.getElementById("sid");
const fresh = document.getElementById("fresh");
```



Abuse of innerHTML

- innerHTML can inject arbitrary HTML content into the page
- However, this is prone to bugs and errors and is considered poor style
  - innerHTML not part of DOM standard
- Best practice: inject plain text only
  - Do not use innerHTML to inject HTML tags

```
// bad style!
var paragraph = document.getElementById("welcome");
paragraph.innerHTML = '<p>text and <a href="page.html">link</a></p>';
```

DOM elements style properties

- Contains same properties as in CSS, but with camelCasedNames
  - examples: backgroundColor, borderLeftWidth, fontFamily

Property	Description
style	Let you set any CSS style property for an element

```
<button id="clickme">Color Me</button>

document.getElementById("clickme").addEventListener("click", changeColor);
function changeColor() {
 const clickMe = document.getElementById("clickme");
 clickMe.style.color = "red";
}
```

Attaching event handlers using JS

- **addEventListener** is a method available on DOM elements that allows you to attach event handlers.
  - The arguments of this method specify what action you want to listen for and what function to execute when that action occurs.

```
<button id="clickme">Color Me</button>

document.getElementById("clickme").addEventListener("click", changeColor);
function changeColor() {
 const clickMe = document.getElementById("clickme");
 clickMe.style.color = "red";
}
```

Anonymous event handlers

- In JavaScript functions can be an anonymous.
  - An anonymous function is a function that does not have a name.

```
const anonymousFunction = function(parameters) {
        Statements
};
```

```
<button id="clickme">Color Me</button>

document.getElementById("clickme").addEventListener("click", function() {
 const clickMe = document.getElementById("clickme");
 clickMe.style.color = "red";
});
```

Common DOM styling errors

- Don’t forget to write .style when setting styles

```
var clickMe = document.getElementById("clickme");
clickMe.color = "red";
clickMe.style.color = "red";
```

- Style properties are capitalized likeThis not like-this

```
clickMe.style.fontSize="14pt";
clickMe.style.font-size="14pt”;
```

- Style properties must be set as strings, often with units at the end

```
clickMe.style.width = 200;
clickMe.style.width = "200px";
clickMe.style.padding = "0.5em";
```

- Write exactly the value you would have written in the CSS, but in quotes

Creating Elements

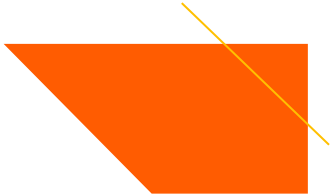
- You can create new elements using **createElement** method
- Append new elements to the DOM object using **appendChild** method

```
<button id="newp">generate a paragraph</button>

document.getElementById("newp").addEventListener("click", generateElement);
function generateElement() {
  const newParagraph = document.createElement("p");
  newParagraph.textContent = "This is a new paragraph";
  document.body.appendChild(newParagraph);
}
```

Removing Elements

- You can remove child elements using **removeChild** method

```
<button id="rmp">remove paragraph</button>

document.getElementById("rmp").addEventListener("click", removeElement);
function removeElement() {
  const elementToRemove = document.getElementById("someId");
  elementToRemove.parentNode.removeChild(elementToRemove);
}
```



Thank You!