EVENTS & EVENT DRIVEN PROGRAMMING

Question:

- What does a webpage do when the page finish loading, but the user hasn’t done anything yet?

Question:

- What does a webpage do when the page finish loading, but the user hasn't done anything yet?

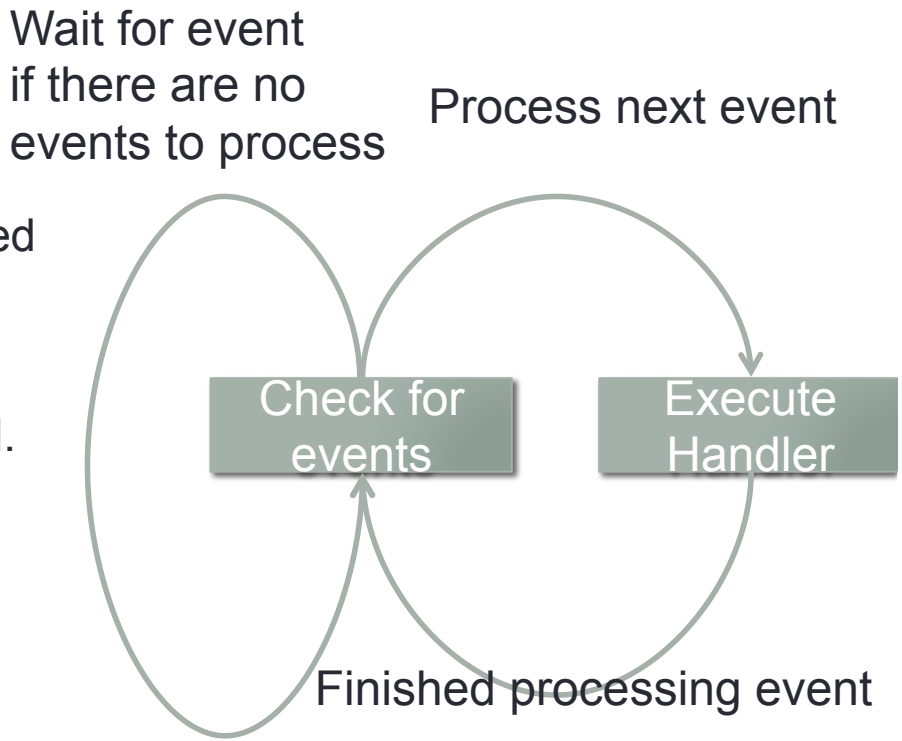
- It waits for input, or other events.
  - Once an event happens then JavaScript code is executed.

Events/Event Driven programming

- Upto now
  - Our programs ran as soon as the page loaded
  - finished running before the use could interact with the page
  - and we used **prompt** and **alert** to interact with the user.
- This is not how a website usually works.
- Website usually
  - waits for user action
  - respond in some way
- This action/response is an Event Driven Programming model.

The event loop

- JavaScript has a built-in event loop.
- After
  - The page finishes loading
  - Any code outside of a function is executed
- JavaScript waits for events
- When an event occurs
  - Adds it to the list of events to be handled.
  - If it's not responding to an event
  1. It looks for an event handler for the earliest even on the event list.
  2. Executes the handler
  3. Removes the event from the list
  4. proceeds to process the next event in the list
  5. Goes back to waiting if there are no more events.



Events/Event Driven programming

- Events
  - An event that is anything that happens on a web-page
    - Significant moments generated by the web-browser
    - Includes any user action
    - Can be and usually is associated with an element.

- We will examine two onload and onclick
  - onload:
    - an event that occurs once all the html file has finished loading.
    - Event associated with the document.
    - Good time to setup our JavaScript
  - onclick
    - an event occurs when a element is clicked. Usually buttons, but any elements.
    - Event associated with the element.

onload

- Responding onLoad :
  - We add an onload attribute to the <body> tag

  - <body **onload**= “<Javascript Code to run>” >
  - </body>

  - Example:

  - <body onload=**“setup()”**>
  - </body>

  - In the above example we call the setup() function after the page loads. Notice that the JavaScript code is inside quotes.
  - When we are in JavaScript, we put HTML in quotes. When we are in HTML we put JavaScript in quotes.

Alternate way of adding Event handler

- `var element = document.getElementById("elementName");`
- `element.onload = functionName;`

- Get the element from the page
- the element will have a property that corresponds to the event, in the above example onload
- assign the function to the event property.
- *Note that the functioName is **not** followed brackets. We don't want to invoke the function, just refer to it.

<input> tag

- We can add inputs to our webpage.
  - For this lab we will add text fields and buttons
  - Both are added with the <input> tag.
- We can use **textfield** rather than **prompts** to accept input in a typical webpage.
- To be able to use textfield we must give the opportunity to type things, this why we needed event driven programming before using textfields.

<input type="text">

- To add a text file we add the tag

```
<input id="input1" type="text">
```

- We can give the tag any id we want.
- but the type is "text"
- In **JavaScript**:
  - We get get the value that the use typed in JavaScript with the value property on the input element.
  - Example:

```
var inputfield = document.getElemebtById("input1");
var userData = inputfield.value
```

<input type="button" onClick="run()">

- We can add buttons to our webpage.

- <input id="button1" type="button" onClick="example()">

- We can give the button any **id**
- The type is equal to "button"
- We have any function run when the button is clicked.
  - We should limit this code to a single function call to keep our code tidy, even though we can place a full JavaScript program between the quotes.
  - This helps keeps our program organized

Other input types

- The <input> tag has other input types that you are free to use but we won't talk about much.
- The basic pattern is the same, you get the value from the input element for most input types, except buttons.
- Examples:
  - color
  - password
  - date
  - file
  - radio
  - range

A simple Example

Persisting information

- If everything is inside functions
- And local variables goes away once after the function executes
- Then information must be persisted in global variable

- If everything is inside functions
  - You can say there is no Main Program
  - Our program is now a collection of functions
  - The functions can run in any order
    - Determined by the user input
    - And changes global variables to keep track of information

Other events

- A list of events that you can respond to can be found here:
- http://www.w3schools.com/jsref/dom_obj_event.asp
- This will be useful for your projects.