# Desarrollo de Aplicaciones Web Empresariales

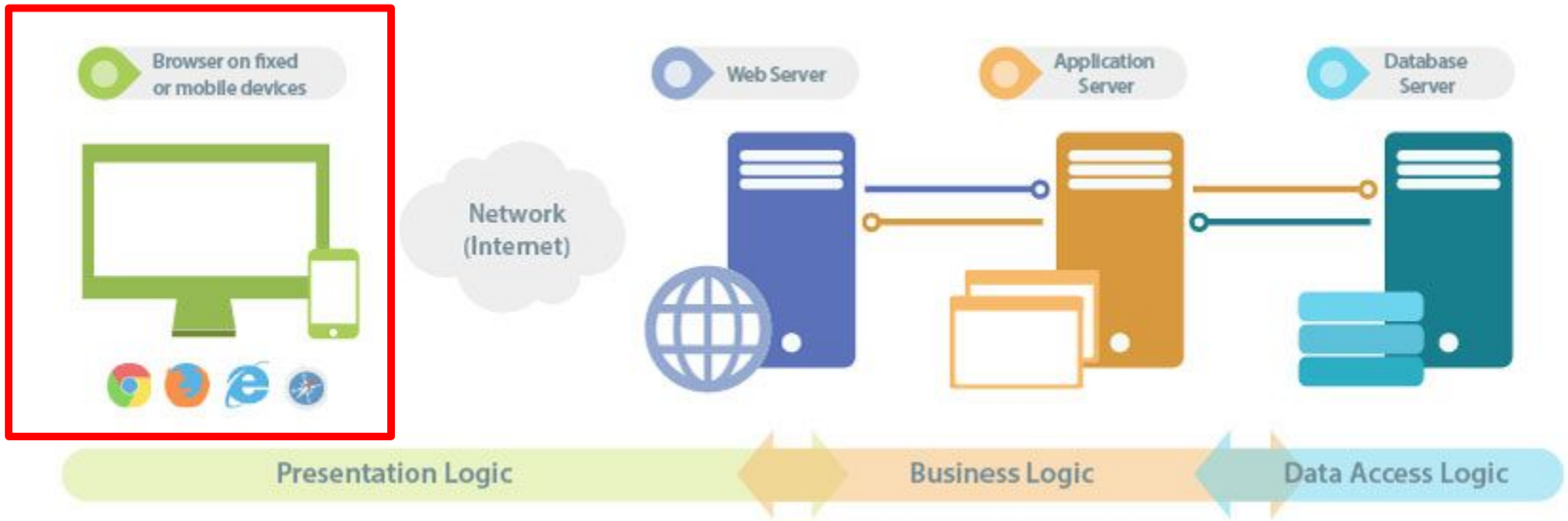
Exequiel Fuentes Lettura
exequiel.fuentes@ucn.cl

# Información de contacto

- Exequiel Fuentes Lettura
  - Email: exequiel.fuentes@ucn.cl
  - Horario de Atención: Jueves y Viernes, bloque C

- Departamento de Ingeniería de Sistemas y Computación
  - Oficina: Y1 - 329
  - http://www.disc.ucn.cl

# Arquitectura de aplicaciones Web: Eventos




Material: https://developer.mozilla.org/en-US/docs/Web/Events

# Eventos: DOM y Javascript

- Tipos de eventos:
  - Mouse: movimiento, click, entrar o salir de un documento.
  - Teclado: abajo, arriba, presionar, etc.
  - Foco: adentro o afuera.
  - Cambios en el elemento **input**, envío de una forma.
  - Eventos Timer.
  - Misceláneos:
    - El contenido de un elemento ha cambiado.
    - Página cargada.
    - Imagen cargada.
    - Excepción no atrapada.

# Manejando eventos

- Para crear un manejador de eventos se debe considerar:
  - Qué sucedió: El evento de interés.
  - Dónde ha sucedido: El elemento de interés.
  - Qué hacer: Se utiliza Javascript para realizar algo.

# Javascript de un elemento

- Opción 1, en el HTML:
  - ...
- Opción 2, desde el Javascript usando el DOM:
  - element.onclick = mouseClick;
  - o
  - element.addEventListener("click", mouseClick);

# El objeto Event

- Las funciones que “escuchan” eventos pasan un objeto Event, típicamente una subclase: MouseEvent, KeyboardEvent, etc.
- Algunas propiedades de Event:
  - type: El nombre del evento ('click', 'mouseDown', 'keyUp', ...).
  - timeStamp: El tiempo en el que el evento fue creado.
  - currentTarget: Elemento que escucha el evento.
  - target: Elemento que lanzó el evento (ejemplo: click sobre algo).

# MouseEvent y KeyboardEvent

- Algunas propiedades de MouseEvent:
  - button: Un botón del mouse se presionó.
  - pageX, pageY: Posición relativa del mouse a la esquina superior izquierda del documento.
  - screenX, screenY: Posición del mouse a la esquina superior izquierda de la pantalla.
- Algunas propiedades del KeyboardEvent:
  - keyCode: Identifica el código de la tecla presionada (deprecada).
  - charCode: Código Unicode de la tecla presionada (deprecada).
  - key: Retorna el nombre que identifica la tecla presionada, ejemplo: “ArrowDown”.

# Cómo mover un elemento - HTML/CSS

```
<style type="text/css">
    #div1 {
        position: absolute;
    }
</style>
...
<div id="div1" onmousedown="mouseDown(event);"
                    onmousemove="mouseMove(event);"
                    onmouseup="mouseUp(event);">
Drag Me!</div>
```

# Cómo mover un elemento - Javascript

```
var isMouseDown = false;  // Dragging?
var prevX, prevY;

function mouseDown(event) {
    prevX = event.pageX;
    prevY = event.pageY;
    isMouseDown = true;
}

function mouseUp(event) {
    isMouseDown = false;
}
```

```
function mouseMove(event) {
    if (! isMouseDown ) {
    return;
    }
    var elem = document.getElementById("div1");
    elem.style.left = (elem.offsetLeft +
        (event.pageX - prevX )) + "px";
    elem.style.top = (elem.offsetTop +
        (event.pageY - prevY )) + "px";
    prevX = event.pageX;
    prevY = event.pageY;
}
```

# Qué manejador se debe usar?

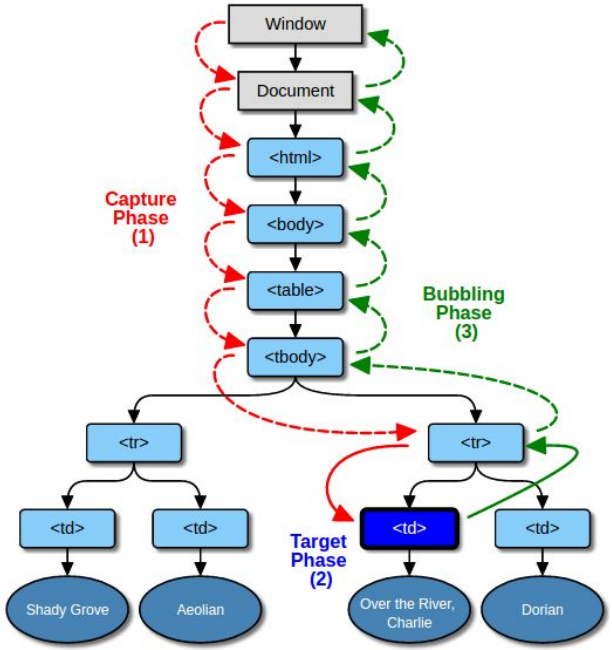
- Elementos pueden contener otros elementos:
  - Suponga que el usuario hace click en “xyz”

```
<body>
   <table>
      <tr>
         <td>xyz</td>
      </tr>
   </table>
</body>
```

- Si tienes manejadores en los elementos td, tr, table y body. Qué manejador se elige?
  - Puede ser el elemento de más adentro.
  - Puede ser el elemento exterior.

# Eventos de captura o de burbuja

- Fase de captura:
  - Comienza desde el elemento de más afuera y propaga hacia los hijos.
  - Un elemento puede parar la propagación, entonces sus hijos no verán el evento.
  - Cómo parar la propagación: event.stopPropagation()
  - element.addEventListener(eventType, handler, **true**);
- Fase de burbuja, usado por la mayoría de los manejadores (ejemplo: onclick)
  - Comienzan desde el elemento que lo genera y se propaga hacia sus ancestros.
  - Cómo parar la propagación: event.stopPropagation()
  - element.addEventListener(eventType, handler, **false**);



# Evento Timer

- Correr una función a partir de 5 segundos desde ahora:
  - token = setTimeout(myFunc, 5*1000);
- Correr una función cada 50 milisegundos:
  - token = setInterval(myfunc, 50);
- Cancelar el Timer: clearInterval(token);
- Se puede utilizar para animaciones, refrescar páginas, etc.

# Concurrencia

- Los eventos son serializados y procesados uno a uno.
- El manejo de eventos no intervienen con otras ejecuciones, es diferente a lenguajes como C o Java.
- Es fácil entender la concurrencia:
  - Los manejadores se ejecutan completamente.
  - No se bloquean.
- Aunque procesos en background es un poco más difícil que utilizando hilos.
- Más información: https://developer.mozilla.org/en/docs/Web/JavaScript/EventLoop

# ¿Preguntas?