Keyboard and Text Events

<table>
<thead>
<tr>
<th>Name</th>
<th>The User Must ...</th>
<th>Applicable Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>blur</td>
<td>remove focus</td>
<td>&lt;a&gt;, &lt;input&gt;, ...</td>
</tr>
<tr>
<td>focus</td>
<td>apply focus</td>
<td>&lt;a&gt;, &lt;input&gt;, ...</td>
</tr>
<tr>
<td>keydown</td>
<td>press a key in an element with focus</td>
<td>&lt;body&gt; and form elements</td>
</tr>
<tr>
<td>keyup</td>
<td>release a key in an element with focus</td>
<td>&lt;body&gt; and form elements</td>
</tr>
<tr>
<td>keypress*</td>
<td>press and release a key in an element with focus</td>
<td>&lt;body&gt; and form elements</td>
</tr>
<tr>
<td>select</td>
<td>select some of the page's text</td>
<td>&lt;input&gt; and &lt;textarea&gt;</td>
</tr>
</tbody>
</table>

* not perfectly supported and not recommended
Debugging JavaScript

- Firefox is good for debugging
  - Select “Web Console” from the Web Developer menu
  - In older versions, select “Error Console” from the Tools menu
  - Displays errors not shown by the browser
- Most web browsers have a developer's console
- Firebug for Firefox provides additional features
  - Can set breakpoints and step through source

What Is DOM?

- Document Object Model
- An application programming interface (API) for (X)HTML
  - Set of objects, properties and methods
  - Allows a programming language to interact with an (X)HTML document via the provided items
- Generally associated with JavaScript
History of DOM

- First proposed by the W3C in the mid-1990s
- DOM 0
  - No formal specification
  - Implemented by all JavaScript-enabled browsers
- Other versions
  - DOM1: 1998
    - First W3C specification
  - DOM2: 2000
    - Mostly supported by Firefox
    - Not really supported by IE
  - DOM3: 2004

Window

- Represents the web browser's window
  - Displaying the current (X)HTML document
- Access via the global object `window`
- The core element in DOM
  - All "global" DOM items actually belong to `window`
  - Both properties and functions
  - Access using `window.item` or simply `item`
    - `e.g., document.write(...) or window.document.write(...)"
## Some Window Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document</td>
<td>Document</td>
<td>The current (X)HTML document</td>
</tr>
<tr>
<td>history</td>
<td>History</td>
<td>The user's recently visited pages</td>
</tr>
<tr>
<td>location</td>
<td>Location</td>
<td>The browser's current URL</td>
</tr>
<tr>
<td>navigator</td>
<td>Navigator</td>
<td>The web browser</td>
</tr>
<tr>
<td>screen</td>
<td>Screen</td>
<td>The user's screen</td>
</tr>
</tbody>
</table>

## Some Window Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alert(...), confirm(...), prompt(...)</td>
<td>Produce dialog boxes</td>
</tr>
<tr>
<td>open(URL, name, ...), close()</td>
<td>Open or close a browser window</td>
</tr>
<tr>
<td>focus(), blur()</td>
<td>Place focus on or take focus away from a browser window</td>
</tr>
<tr>
<td>moveBy(x, y), moveTo(x, y)</td>
<td>Change the browser window's position</td>
</tr>
<tr>
<td>resizeBy(w, h), resizeTo(w, h)</td>
<td>Change the browser window's size</td>
</tr>
<tr>
<td>scrollBy(x, y), scrollTo(x, y)</td>
<td>Scroll within the browser window</td>
</tr>
</tbody>
</table>
Document

- The actual (X)HTML document and all of its elements

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>Body</td>
<td>The document's body</td>
</tr>
<tr>
<td>title</td>
<td>String</td>
<td>The title of the document</td>
</tr>
<tr>
<td>anchors[],</td>
<td>Array</td>
<td>Arrays containing all of the</td>
</tr>
<tr>
<td>images[]</td>
<td></td>
<td>document's anchors, images, ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getElementById(id)</td>
<td>Element</td>
<td>The element with the given id</td>
</tr>
<tr>
<td>getElementsByName(name)</td>
<td>Array</td>
<td>All elements with the given name</td>
</tr>
<tr>
<td>getElementsByTagType(tagName)</td>
<td>Array</td>
<td>All elements with the given tagName</td>
</tr>
</tbody>
</table>

Elements in DOM

- Every element on a page is associated with an HTMLElement object in the DOM
- Objects are most easily accessed with the id attribute

```html
<tag id="idValue"> ... </tag>
```

```javascript
document.getElementById("idValue")
```

- What can you do with an HTMLElement?
  - Change its content
  - Change its style
  - Change element-specific attributes (e.g., src, href)
**Some HTMLElement Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>className</td>
<td>String</td>
<td>Value(s) for the class attribute</td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>Value of the id attribute</td>
</tr>
<tr>
<td>innerHTML</td>
<td>String</td>
<td>All content between the element's opening and closing tags</td>
</tr>
<tr>
<td>style</td>
<td>Style</td>
<td>Contains the element's style properties</td>
</tr>
<tr>
<td>tagName</td>
<td>String</td>
<td>The HTML tag of the element (uppercase)</td>
</tr>
</tbody>
</table>

**Style**

- The CSS properties and values applied to an element
- Style properties correspond to CSS properties
  - Style properties don't use hyphens
    - *e.g.*, `style.backgroundColor` ↔ `background-color`
  - Style property values are Strings
    - *e.g.*, `style.width` → "100px"
- You can change the style by changing these values
  - *e.g.*, `style.color` = "blue"
Dynamic Elements

- To change the appearance of an element ...
  - use the `style` property of all DOM elements
- To change the text in an element ...
  - use the `innerHTML` property of all DOM elements
- To change the position of an element ...
  - set `position` to `absolute` or `relative` in CSS
    - use the `style` property to change the offset
      - `top`, `right`, `bottom` and `left`
- To make an element appear or disappear ...
  - change the `style`'s `visibility` or `display`

Unobtrusive JavaScript

- Goal: separate content, style, and behaviour
  - Content in (X)HTML
  - Style in separate CSS file
  - Behaviour in separate JavaScript file
- Event handlers in tags → lack of separation
  - Better to `dynamically attach` them in JavaScript
- Every DOM element has properties that match events

```javascript
domElement.onEvent = functionName;
```

- Get the element from the document
- Use these properties to attach event handlers
  - `e.g., document.getElementById("id").onEvent = functionName;`
Attaching Handlers

- **Via `onload` for Window**

  ```javascript
  window.onload = attachHandlers;
  ```

  ```javascript
  attachAndHandle
  function handler1() {
  ...
  }
  ...
  function attachHandlers() {
  document.getElementById("id1").onclick=handler1;
  document.getElementById("id2").onchange=handler2;
  }
  window.onload = attachHandlers;
  ```

- **In a second JavaScript file (not ideal)**

  ```javascript
  <script src="handleEvents.js"></script>
  ```

  ```javascript
  handleEvents
  function handler1() { ... }
  function handler2() { ... }
  ...
  ```

  ```javascript
  attachHandlers
  document.getElementById("id1").onclick=handler1;
  document.getElementById("id2").onchange=handler2;
  ```

  Must call after elements have been created (e.g., after `</body>` tag)