null and undefined

- null is synonymous with nothing
  - e.g., no value, nothing there
- undefined is synonymous with confusion
  - e.g., what's this?
  - All declared variables have undefined as an initial value ...
    - ... until another value is assigned
Some Handy Things

- **typeof**
  - Operator that determines the type of a parameter
  - Returns the type name as a String
  - Can use `typeof item` or `typeof(item)`

- **Date**
  - Object with timing functions
  - `var localTime = new Date();`

<table>
<thead>
<tr>
<th>A Few Functions</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>toLocaleString()</code></td>
<td>The current date and time as a String</td>
</tr>
<tr>
<td><code>getDay()</code>, <code>getMonth()</code>, <code>getDate()</code>, <code>getFullYear()</code></td>
<td>The current day of the week, month, day of the month and year</td>
</tr>
<tr>
<td><code>getTime()</code></td>
<td>The number of ms since Jan. 1, 1970</td>
</tr>
</tbody>
</table>

Dialogs

- **alert**(message)**
  - Displays `message` with an **OK** button

- **confirm**(message)**
  - Displays `message` with **OK** and **Cancel** buttons
    - Returns **true** for **OK**, **false** for **Cancel**

- **prompt**(promptMsg, sampleText)**
  - Displays `promptMsg` with **OK**, **Cancel** and a text input box
    - `sampleText` is optional; it is displayed in the input box
    - Returns input for **OK**, **null** for **Cancel**
true and false

- Boolean true and false are truly so
- What is falsey?
  - Something that evaluates to false but isn't really
    - String "" (empty)
    - Number 0
    - NaN, null and undefined
- What is truthy?
  - Something that evaluates to true but isn't really
    - All strings except ""
    - All numbers except 0

Boolean Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>x &amp;&amp; y</td>
<td>x ∧ y</td>
<td>x AND y</td>
</tr>
<tr>
<td>!x</td>
<td>¬x</td>
<td>NOT x</td>
</tr>
</tbody>
</table>

- Only use Boolean operands
  - (x || y) yields x if x is truthy; it yields y otherwise
  - (x && y) yields x if x is falsey; it yields y otherwise
Testing Equivalence

<table>
<thead>
<tr>
<th>Operator</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;, &gt;=</td>
<td>Greater than (or equal to)</td>
</tr>
<tr>
<td>&lt;, &lt;=</td>
<td>Less than (or equal to)</td>
</tr>
<tr>
<td>!=, ==</td>
<td>(Not) equal to</td>
</tr>
<tr>
<td>!==, ===</td>
<td>Strictly (not) equal to</td>
</tr>
</tbody>
</table>

- Standard equivalence operators performs conversion
  - *e.g.*, 4 == "4" -> true

- Strict equivalence operators take type into account
  - *e.g.*, 4 === "4" -> false
Functions

- Function called with too many parameters?
  - Extra parameters are ignored
  - Can access full list using `arguments` array

- Function called with too few parameters?
  - Others are assigned values of `undefined`

- What if no return value is specified?
  - Function returns `undefined`
Variable Scope

- **Local variables**
  - Explicitly declared inside a function
  - Can only be used within that function

- **Global variables**
  - Declared outside a function
    - Implicit or explicit
  - Implicitly declared inside a function
  - Can be used in any subsequent function or statement

```javascript
var x = 1;

function a()
{
    var y = 2;
    z = 4;
    ...
}

w = 8;
...
```

Arrays

- There are several ways to create an array

```javascript
var anArray = [];
anArray[0] = 12;
anArray[1] = "hello";
anArray[2] = (3 / 4);
```

Or

```javascript
var anArray = new Array(1);
anArray[0] = 12;
anArray[2] = (3 / 4);
anArray[1] = "hello";
```

Or

```javascript
var anArray = [12, 0.75, "hello"];
```

Or

```javascript
var anArray = new Array(12, 0.75, "hello");
```

- **Array size is dynamic**
  - Increases to accommodate largest index
  - Some entries may be undefined
Some Array Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>concat(array1, ... , arrayN)</code></td>
<td>A single array formed from the parameters</td>
</tr>
<tr>
<td><code>push(value1, ... ,valueN)</code></td>
<td>Adds the given value(s) to the end of an array</td>
</tr>
<tr>
<td><code>unshift(value1, ... ,valueN)</code></td>
<td>Adds the given value(s) to the front of an array</td>
</tr>
<tr>
<td><code>shift()</code></td>
<td>Removes the first element of the array and returns it</td>
</tr>
<tr>
<td><code>sort()</code></td>
<td>Sorts the elements in an array</td>
</tr>
</tbody>
</table>

Best Practices

- Use semicolons to end statements
  - They may be optional ...
  - ... but they may cause issues if left out
- Explicitly declare all variables
- Avoid using global variables
- Comments are always important
  - Use `//` for single lines
  - Use `/* ... */` for multiple lines
Events

- Browsing requires user interaction
  - Move and click the mouse
  - Load and leave websites
  - Type in text boxes
- These actions are perceived as events
  - Browser responds to such events
- HTML 4 defined a series of such events
  - 18 in total

JavaScript and Events

- JavaScript can hook into browser events
  - Create functions to respond to user actions
    - Called event handlers
  - Register the functions to events and elements
    - Can be done in the HTML file

```
<tag oneventName = "functionName()">...
</tag>
```

- functionName is called when eventName occurs for the given element
- This is an example of event-driven programming
## Document Events

<table>
<thead>
<tr>
<th>Name</th>
<th>The Document Must ...</th>
<th>Applicable Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>load</td>
<td>finish loading</td>
<td>&lt;body&gt;</td>
</tr>
<tr>
<td>unload</td>
<td>be left by the user</td>
<td>&lt;body&gt;</td>
</tr>
</tbody>
</table>

## Mouse Events

<table>
<thead>
<tr>
<th>Name</th>
<th>The User Must ...</th>
<th>Applicable Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>click</td>
<td>click on the element</td>
<td>Most</td>
</tr>
<tr>
<td>dblclick</td>
<td>double-click on the element</td>
<td>Most</td>
</tr>
<tr>
<td>mouseover</td>
<td>move the mouse over the element</td>
<td>Most</td>
</tr>
<tr>
<td>mousemove</td>
<td>move the mouse within the element</td>
<td>Most</td>
</tr>
<tr>
<td>mouseexit</td>
<td>move the mouse away from the element</td>
<td>Most</td>
</tr>
<tr>
<td>mousedown</td>
<td>press the mouse button</td>
<td>Most</td>
</tr>
<tr>
<td>mouseup</td>
<td>release the mouse button</td>
<td>Most</td>
</tr>
</tbody>
</table>