Browser Object Model

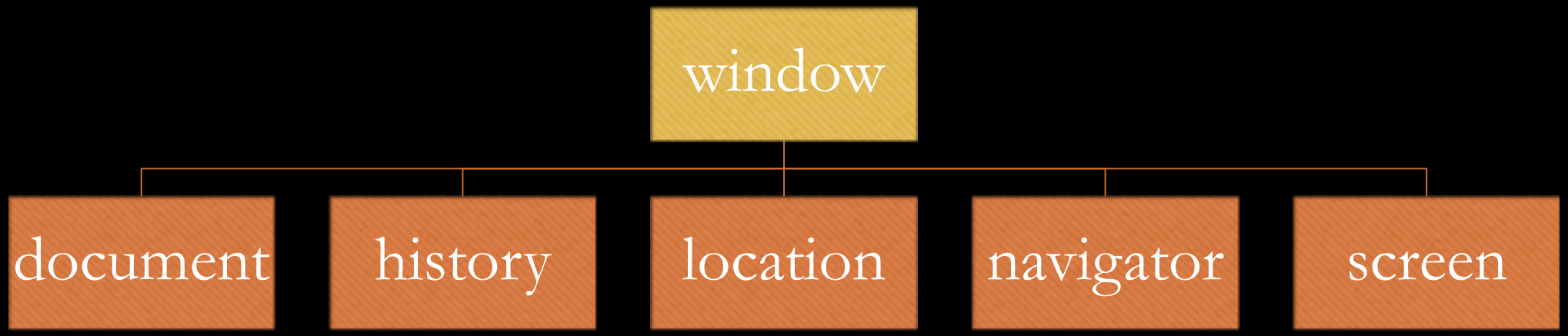
BOM

User Interfaces

420-WC4-AB

The window Object

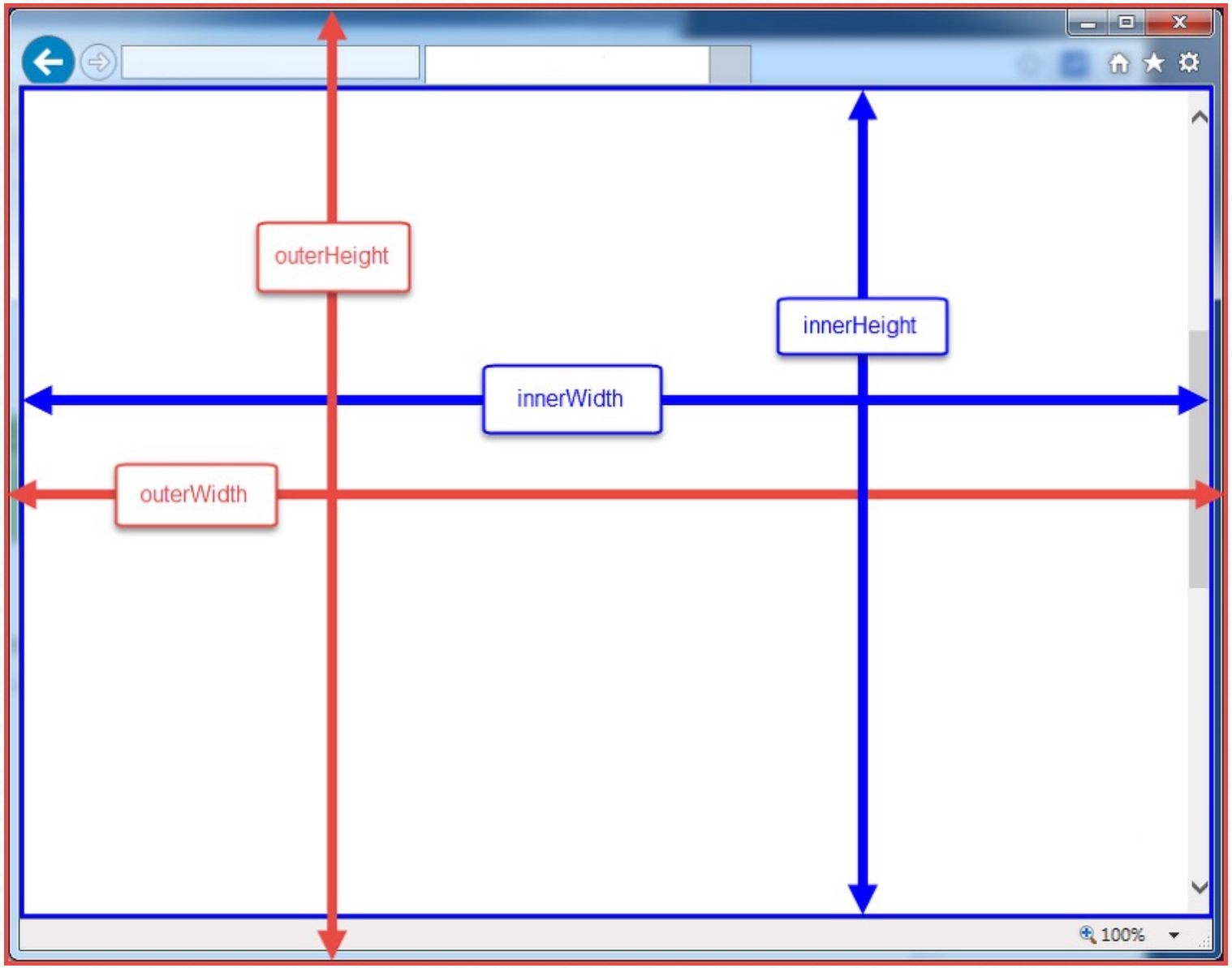
Window is the object of browser, it is the global object of JavaScript



Window Object Properties

- Properties include

```
window.innerHeight; // inner height of the browser in pixels
window.innerWidth;  // inner width of the browser in pixels
window.outerHeight; // height of the entire browser in pixels
window.outerWidth;  // width of the entire browser in pixels
// .. ETC
```


outerHeight
innerHeight
innerWidth
outerWidth
100%

Window Methods

- Methods include

```
window.open( ... ); // open a new window or tab
window.resizeTo( ... );  // resize current window to size
window.resizeBy( ... ); // resize current window by amount
window.moveTo( ... );  // move current window to location
window.moveBy( ... ); // move current window by amount
window.close( ... );  // close current window
```

The Navigator Object

- The Navigator object is used for browser detection and getting information about the browser.
- Read-only properties.
- Different browser provides different capabilities which are not standardized

```
navigator.appCodeName;   //returns the code name
navigator.appName;       // returns the name
navigator.appVersion;    // returns the version
navigator.cookieEnabled;  // returns Boolean cookies are enabled
navigator.language;      // returns the language
navigator.userAgent;     // returns the user agent
navigator.platform;      // returns the platform
navigator.onLine;        // returns boolean - browser is online
// ETC ...
```

The Screen Object

- The Screen allows you to gather information about the visitor's screen

```
screen.availHeight;
//Returns the height of the screen available to browser

screen.availWidth;
//Returns the width of the screen available to browser

screen.height;
// Returns the height of the screen in pixels

screen.width;
// Returns the width of the screen in pixels

// ETC...
```

The History Object

- The history object represents an array of URLs visited by the user.
- By using this object, you can see how many you can load previous, move to or access particular pages.
- For the security reason, you can only navigate the history without knowing exact URLs

```
history.length;    // returns the length of history URLs
history.forward();   // loads the next page
history.back();    // loads the previous page
history.go( x );     // loads the X page of your history
```

-

The Location Object

- The Location allows you to gather information about the current page address URL

```
window.location.href
// returns the href (URL) of the current page
// assign it a value and it will redirecct the page

window.location.hostname
/ returns the domain name of the web host

window.location.protocol
// returns the web protocol used (http: or https:)

window.reload()
// reloads/refresh the current page

// ETC ...
```

alert(myVar);

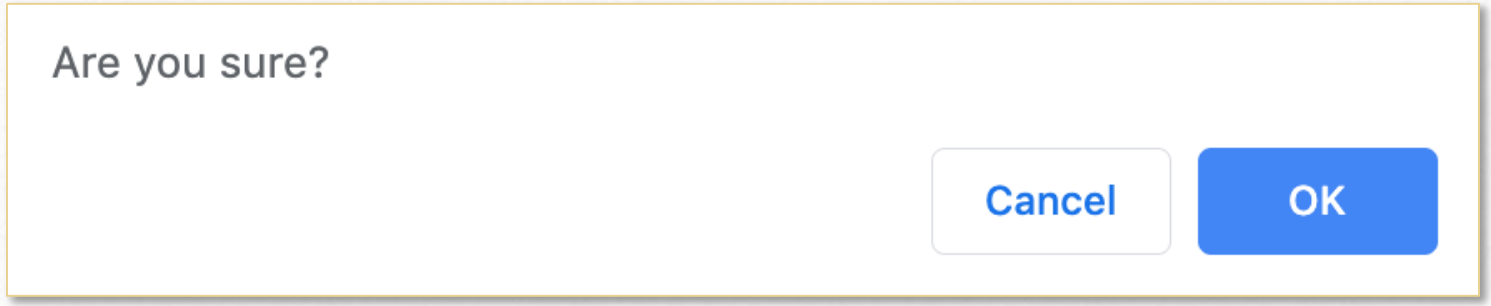
- Alert displays the alert box containing the message in *myVar* with ok button

```
alert("Hello World");
```

confirm(myVar);

- Confirm displays the confirm dialog box containing *myVar* with ok and cancel button

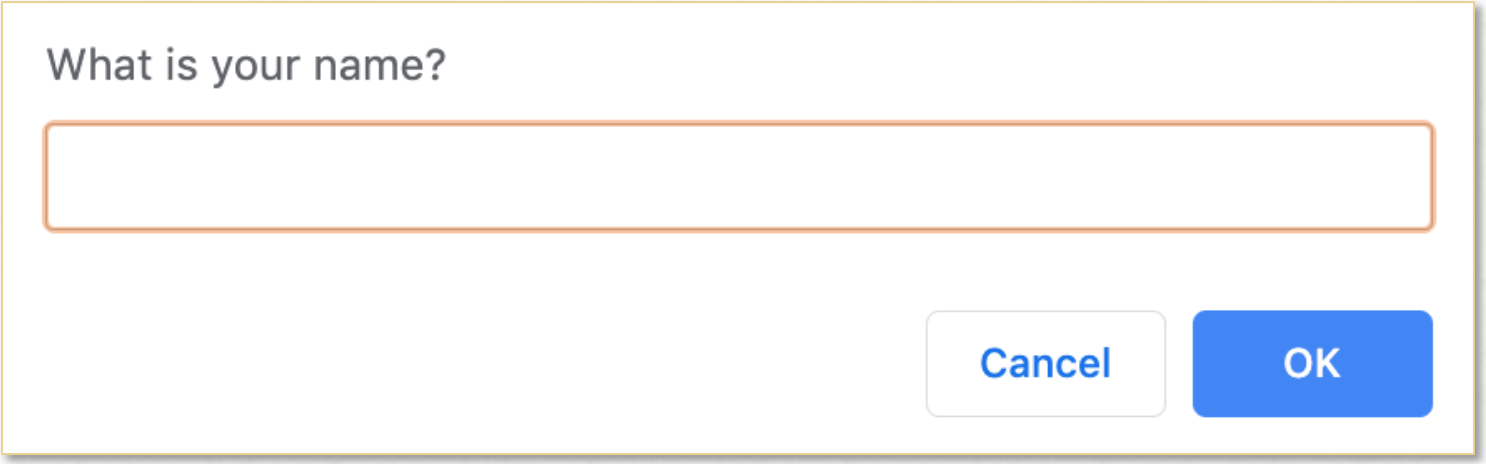
```
let answer = confirm("Are you sure?");
```



prompt(myVar);

- Prompt displays a dialog box containing *myVar* with text input box, ok and cancel button. Returns *null* is cancel is pressed.

```
let answer = prompt("What is your name?");
```



Timers

Scheduling function calls

- Scheduling a function call allows you to execute a function at a later time.
- There are two functions that help us schedule a function call for later:
  - **setTimeout** allows to run a function once after the interval of time.
  - **setInterval** allows to run a function regularly with the interval between the runs.

setTimeout

- setTimeout allows to execute a function once after a set interval of time.
- It requires 2 arguments:
  - function – the callback function to be executed after the specified delay
  - delay - how long to wait before the function gets executed (in milliseconds)
- The function does return a variable, it can be useful for cancelling timers.

```
let timerId = setTimeout( function, delay )
```

setTimeout() example

- This will call the function sayHello after 1000 milliseconds (1 seccond)

```
let timerId = setTimeout(sayHello, 1000);
function sayHello() {
  alert( "Hello World" );
}
```

setTimeout() example

- This will call the function sayHi along with the 2 parameters "Hello" and "John" after 1000 milliseconds (1 seccond)

```
let timerId = setTimeout(sayHi, 1000, "Hello", "John");
function sayHi(phrase, who) {
  alert( phrase + ', ' + who );
}
```

clearTimeout()

- To cancel a timeout we use the *timerID* that is returned when we create our timers as the argument of a clearTimeout() function

```
clearTimeout( timerId );
```

setInterval

- setInterval allows to run a function regularly with a interval between execution of each function call.
- It requires 2 arguments:
  - function - the callback function to be executed after the specified delay
  - delay - how long to wait before the function gets executed (in milliseconds)
- The function does return a variable, it can be useful for cancelling timers.

```
let timerId = setInterval( function, delay )
```

setInterval() example

- This will call the function sayHi along with the 2 parameters "Hello" and "John" every 1000 milliseconds (1 seccond)

```
let timerId = setInterval(sayHi, 1000, "Hello", "John");
function sayHi(phrase, who) {
  alert( phrase + ', ' + who );
}
```

clearInterval()

- To cancel an interval we use the *timerID* that is returned when we create our timers as the argument of a clearInterval() function

```
clearInterval( timerId );
```

Questions ?