

# JavaScript

---

Dr. Garrett Dancik

# JavaScript overview

- JavaScript is a programming language that can be used to add, remove, change, or modify HTML elements and CSS settings on a web page.
- JavaScript can also be used as a standalone language, but most often is it used for creating *dynamic* web pages
- JavaScript syntax has many similarities with C++ and Java, though it is not related to either
- These notes and examples highlight some of the key concepts and differences between JavaScript and other programming languages.

# First JavaScript example

*// example for loop*

```
let sum = 0;
```

```
for (let i = 1; i <= 10; i++) {
```

```
    sum += i;
```

```
}
```

```
document.write('<p>The sum of 1-10 is: ' + sum + '</p>');
```

*// example if..else statement*

```
sum = 0;
```

```
if (sum > 5) {
```

```
    document.write('<p>The sum is greater than 5</p>');
```

```
} else {
```

```
    document.write('<p>The sum is NOT greater than 5</p>');
```

```
}
```

# JavaScript variable declaration and scope

- JavaScript **best practices** for declaring variables:
  - Use *var* only for global variables (though only use global variables if necessary)
  - Use *const* for constant variables (elements of arrays can still be changed) (introduced 2015)
  - Use *let* for all other variables (introduced 2015)

Declaration	Example	Scope	Allows redeclaration?	Allows reassignment?
<i>const</i>	<code>const x = 4;</code>	Block	No	No
<i>var</i>	<code>var x = 4;</code>	Global or Function	Yes	Yes
<i>let</i>	<code>let x = 4;</code>	Block	No	Yes
<i>(none)*</i>	<code>x = 4;</code>	Global	Yes	Yes

\*When using *strict* mode, referencing an undefined variable will cause an error. You should always use *strict* mode! ([https://www.w3schools.com/js/js\\_strict.asp](https://www.w3schools.com/js/js_strict.asp))

# Variable declaration and scope (con't)

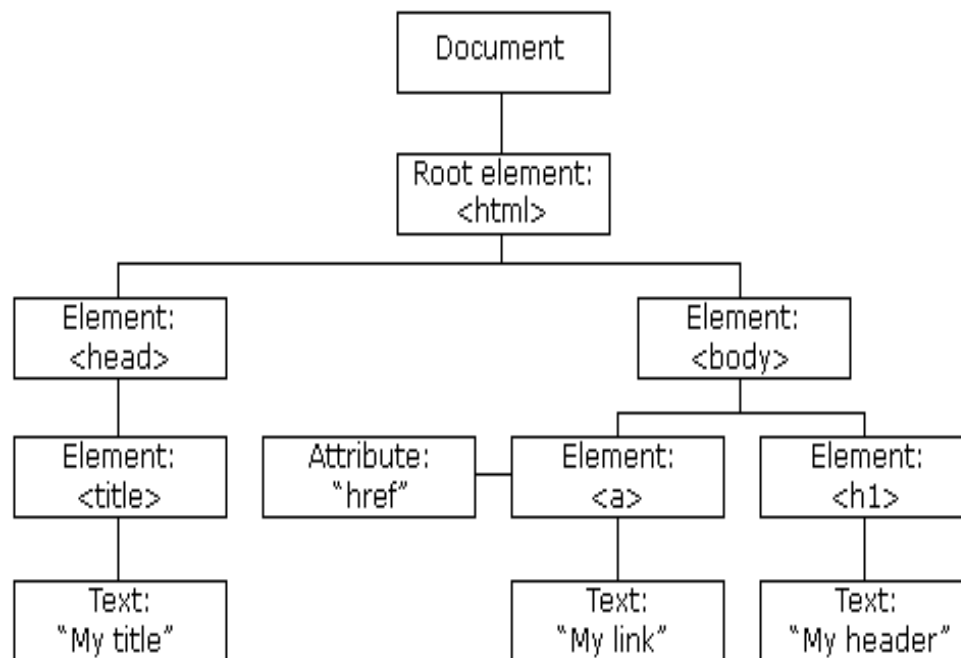
- A variable that is not initialized will have the value *undefined*
- JavaScript has a bizarre behavior known as “hoisting” where variable declarations but not assignments are moved to the top of the current scope
  - This also allows you to call functions before they are defined, if a *function declaration* is used
  - Variables declared using *let* and *const* are technically hoisted but cannot be accessed until they are initialized
- More details and examples:
  - [https://www.w3schools.com/js/js\\_scope.asp](https://www.w3schools.com/js/js_scope.asp)
  - [https://www.w3schools.com/js/js\\_let.asp](https://www.w3schools.com/js/js_let.asp)

# HTML Document Object Model (DOM)

- The HTML DOM provides standards for programmatically accessing, changing, adding, or deleting HTML elements

- Key observation:

- The DOM defines a tree where HTML elements have *children* and *parents*
- Each HTML element has attributes and styles and includes its children



# Finding and changing HTML elements

*These work for any element (not just the document)*

Method For Finding HTML elements	Description
<code>document.getElementById(id)</code>	Find an element by its unique id (returns a <b>single</b> element)
<code>document.getElementsByTagName(name)</code>	Find elements by tag name (returns a <b>HTMLCollection</b> )
<code>document.getElementsByClassName(name)</code>	Find elements by class name (returns a <b>HTMLCollection</b> )

Syntax for accessing and/or changing* an element	Description
<code>element.innerHTML</code>	The inner HTML of an element (may contain HTML tags)
<code>element.innerText</code>	The inner text of an element (HTML tags are ignored)
<code>element.attribute</code>	The attribute value of an HTML element
<code>element.style.property</code>	The style of an HTML element (properties are in camelCase, e.g., 'background-color' is 'backgroundColor')

\*Assignment is used to change the corresponding value; for example to change the HTML of an element use, e.g., `element.innerHTML = "<h2> Changed </h2>"`

Modified from: [https://www.w3schools.com/js/js\\_htmldom\\_document.asp](https://www.w3schools.com/js/js_htmldom_document.asp)

# Functions in JavaScript

- Functions can be created using a *function declaration*:
  - ```
function add(x,y) {  
    return x + y;  
}
```
- Functions can be created using a *function expression*:
  - ```
myfunction = function(x,y,...) {  
    return x + y;  
}
```
- *Arrow functions* are a short-hand way of defining function expressions, by omitting the *function* and *return* keywords, and the braces:
  - ```
myfunction = (x,y) => x + y;
```



# Functions in JavaScript (con't)

- The *this* keyword refers to the object the code belongs to.
  - By itself, *this* refers to the [Global Window]
  - In a function declaration or expression, *this* refers to the element that called the function
    - If the function was called from the window and you are in strict mode, then *this* is undefined; otherwise, *this* is [object Window]
  - For arrow functions, *this* refers to the *owner* of the function
    - Don't use arrow functions to respond to events if you need to know what triggered the event
- Function declarations are *hoisted* while function expressions and arrow functions are not.
- For more information, see:
  - [https://www.w3schools.com/js/js\\_function\\_definition.asp](https://www.w3schools.com/js/js_function_definition.asp)
  - [https://www.w3schools.com/js/js\\_this.asp](https://www.w3schools.com/js/js_this.asp)

# Events in JavaScript

| Event       | Description                                        |
|-------------|----------------------------------------------------|
| onchange    | An HTML element has been changed                   |
| onclick     | The user clicks an HTML element                    |
| onmouseover | The user moves the mouse over an HTML element      |
| onmouseout  | The user moves the mouse away from an HTML element |
| onkeydown   | The user pushes a keyboard key                     |
| onload      | The browser has finished loading the page          |

Table from: [https://www.w3schools.com/js/js\\_events.asp](https://www.w3schools.com/js/js_events.asp)

Complete list: [https://www.w3schools.com/jsref/dom\\_obj\\_event.asp](https://www.w3schools.com/jsref/dom_obj_event.asp)

# Events in JavaScript

- You can set the event attribute of the element:
  - `<button onclick = "alert('hi');">Click </button>`
    - The value of an attribute is a string that specifies JavaScript to be executed
- You can use an event listener to attach or remove an event handler to an element:
  - `element.addEventListener("click", myfunction);`
  - `element.removeEventListener("click", myfunction);`
  - The second argument of `addEventListener` must be a *function*, though it can be anonymous, but anonymous functions can't be removed:
    - `element.addEventListener("click", function() {alert('hi');});`
- Using the event listener gives you more control and is preferred as it separates the web page structure from its logic, and allows the event to trigger multiple function calls
- More information
  - [https://www.w3schools.com/js/js\\_events.asp](https://www.w3schools.com/js/js_events.asp)
  - [https://www.w3schools.com/js/js\\_htmlDOM\\_eventlistener.asp](https://www.w3schools.com/js/js_htmlDOM_eventlistener.asp)

# Additional notes

- Most of the time for web development, we use JavaScript in order to:
  - get one or more HTML elements (by id, class name, tag name, etc) in order to change the HTML or attributes of an element
  - respond to events (clicking, mousing over an element, etc)
- We will use JavaScript arrays and objects (see the examples)
- Tips for troubleshooting code:
  - Look at the console (which is where errors will be displayed)
    - Generally, there will not be errors on the page.
  - Use *console.log* and alerts to test your code. These can be especially useful to test whether a function is being called.