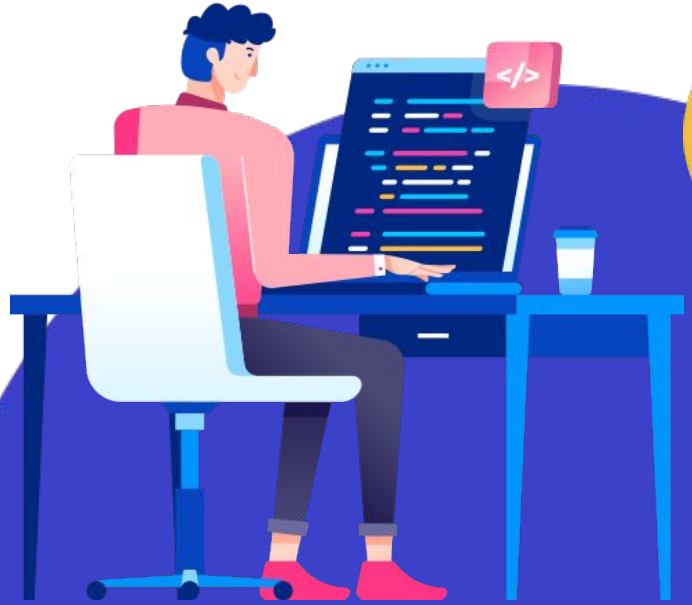
PROGRAMMING FUNDAMENTALS

JS Fundamentals 2

DIPLOMA IN FULL-STACK DEVELOPMENT

Certificate in **Computing Fundamentals**



TODAY'S AGENDA

- Recap
- DOM
- Event Listener
- Event Objects

Recap

JavaScript Basics
Data Types

Variables

* 3 different keywords

Var
- Function scoped
- can be changed in scope
- Avail. before declared!

let
- Block scoped
- can be changed in scope
- Only avail after declaration

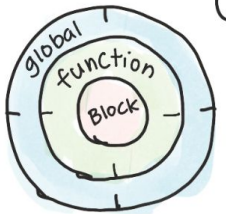
Const
- Block scoped
- cannot be changed
- Only avail after declaration

keyword — variable name

```
const greeting = "Hello";
```

Declaring a variable — assigned value

scope



Data types

♥ String

a set of characters that reside between single or double quotes.

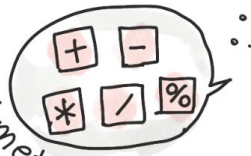
♥ Number

```
let donut = 32;
```

can be: integer, negative decimals, etc.

also: Infinity, BigInt

Arithmetic operators



- + Addition
- − Subtraction
- * multiplication
- / division
- % Remainder

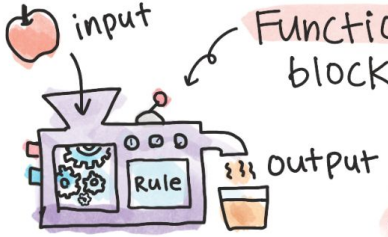
♥ Boolean

true

false

@AzureAdvocates / @girlie_mac

JavaScript Basics
Functions



Function: a building block of code that we can execute on demand.

- Take an input
- Return an output

Declaration

```
function square(n) {
  return n * n;
}
```

function name

parameter(s)

Return an output

Passing info

```
function juice {
  (apple, orange)
}
```

When calling the function, you'll store the value in a variable

```
Const myNum
  = square(25);
```

Dafault values

```
function displayGreet (name, sal='Hello') {
  console.log(`${sal}, ${name}`);
}
```

Function as parameter

```
function displayDone( ){
  Console.log('3 sec. elapsed.');
}
SetTimeout(3000, displayDone)
```

Rewrite

Anonymous Function

```
SetTimeout(3000, function(){
  console.log(---- );
}
```

Rewrite

ES6

Fat Arrow Function

```
setTimeout (3000, () => {
  console.log( ---- );
}
```

@AzureAdvocates / @girlie_mac

JavaScript Basics
Making Decisions

Booleans

true false

```
let myStatement = true
let anotherStatement = false
```

If statements

if else

```
if (status === 200) {
  message = 'OK';
} else {
  message = 'Error!';
```

true → message = 'OK'; condition

false → message = 'Error!';

Ternary

```
const message =
  (status === 200) ? 'OK' : 'Error!';
```

Comparing values

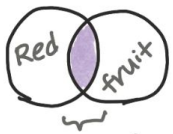
>	Greater than	<	Less than
>=	Greater than or equal to	<=	Less than or equal to
==	equal values	!=	not equal values
===	equal values + data types	!==	not equal values + data types

```
'42' == 42   true
'42' === 42  false
```

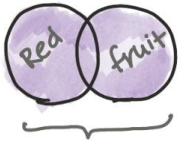
String v. number

Logical Operators

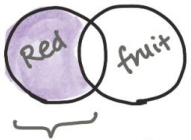
&&	||	!
logical AND	logical OR	logical NOT



Red AND fruit



red OR fruit

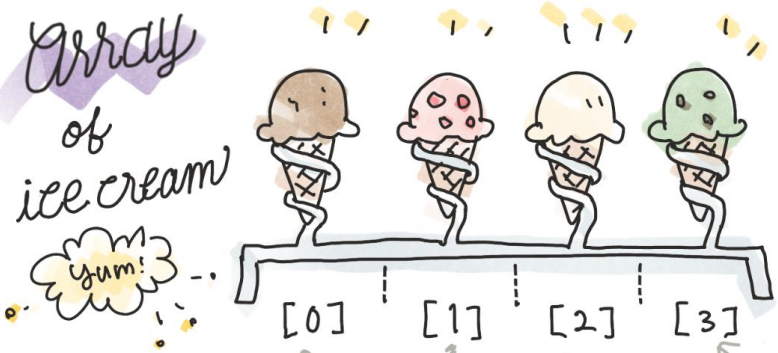


red, not fruit

@AzureAdvocates / @girlie_mac

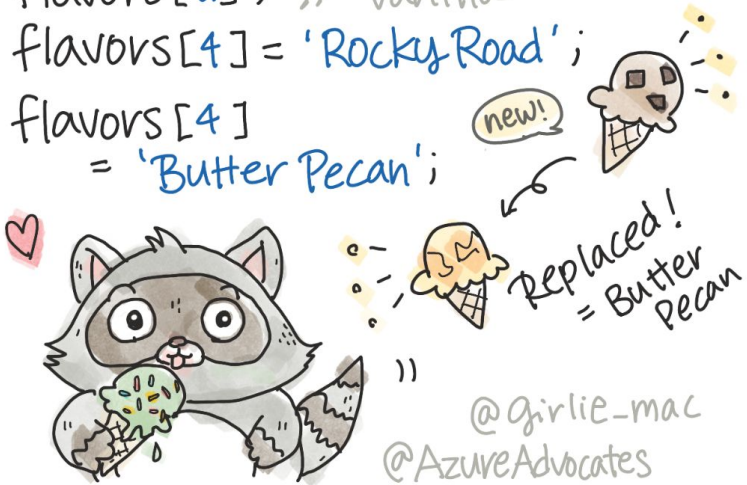
JavaScript Basics
Arrays [and Loops]

Array of ice cream




```
let flavors =
['chocolate', 'strawberry', 'Vanilla',
 'Pistachio'];

flavors[2]; // 'Vanilla'
flavors[4] = 'Rocky Road';
flavors[4]
  = 'Butter Pecan';
```



@girlie_mac
@AzureAdvocates

Loops

For Loop

```
for (let i = 0; i < flavors.length; i++){
  console.log (flavors[i]);
}
```

prints out each flavor after each iteration!

While-Loop

```
let i = 0;
while (i < flavors.length) {
  console.log (flavors[i]);
  i++;
}
```

The loop will stop when the condition is met!

DOM

Document Object Model

DEFINITION

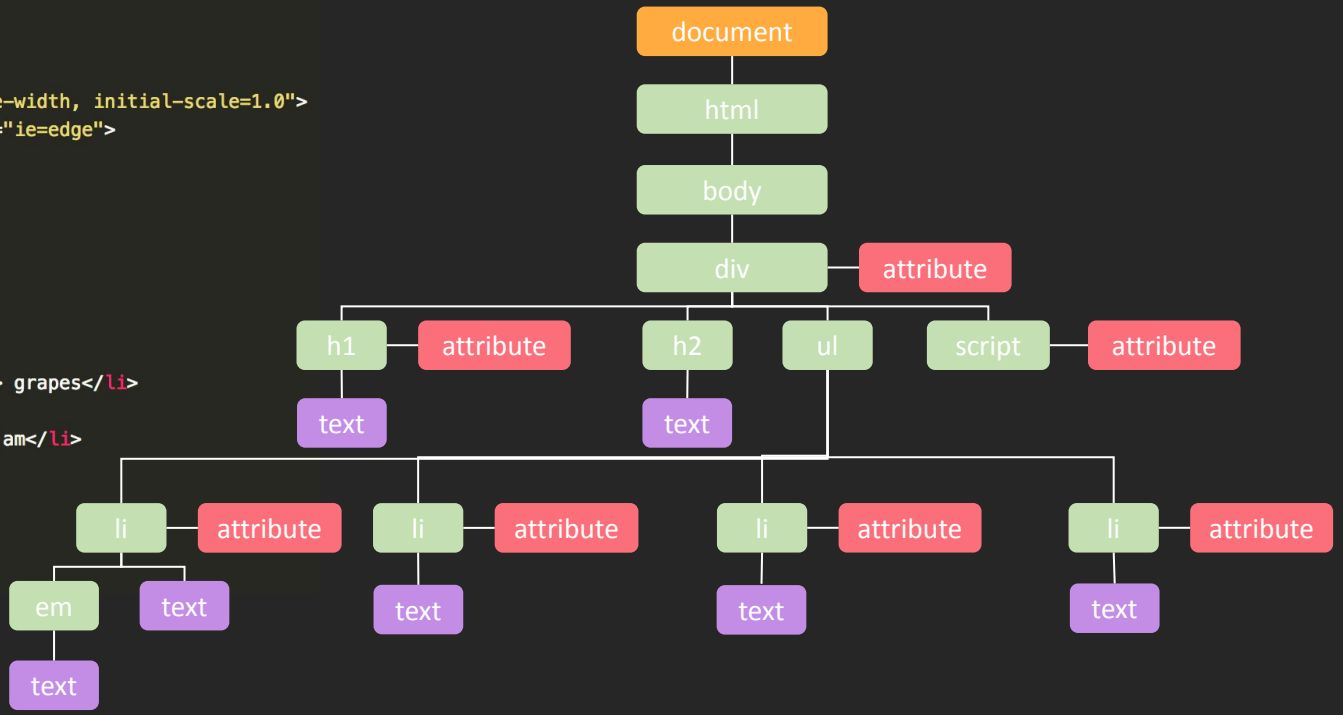
As a browser loads a webpage, it creates a **model** of that page. The model is called a **DOM Tree**

SAMPLE HTML PAGE

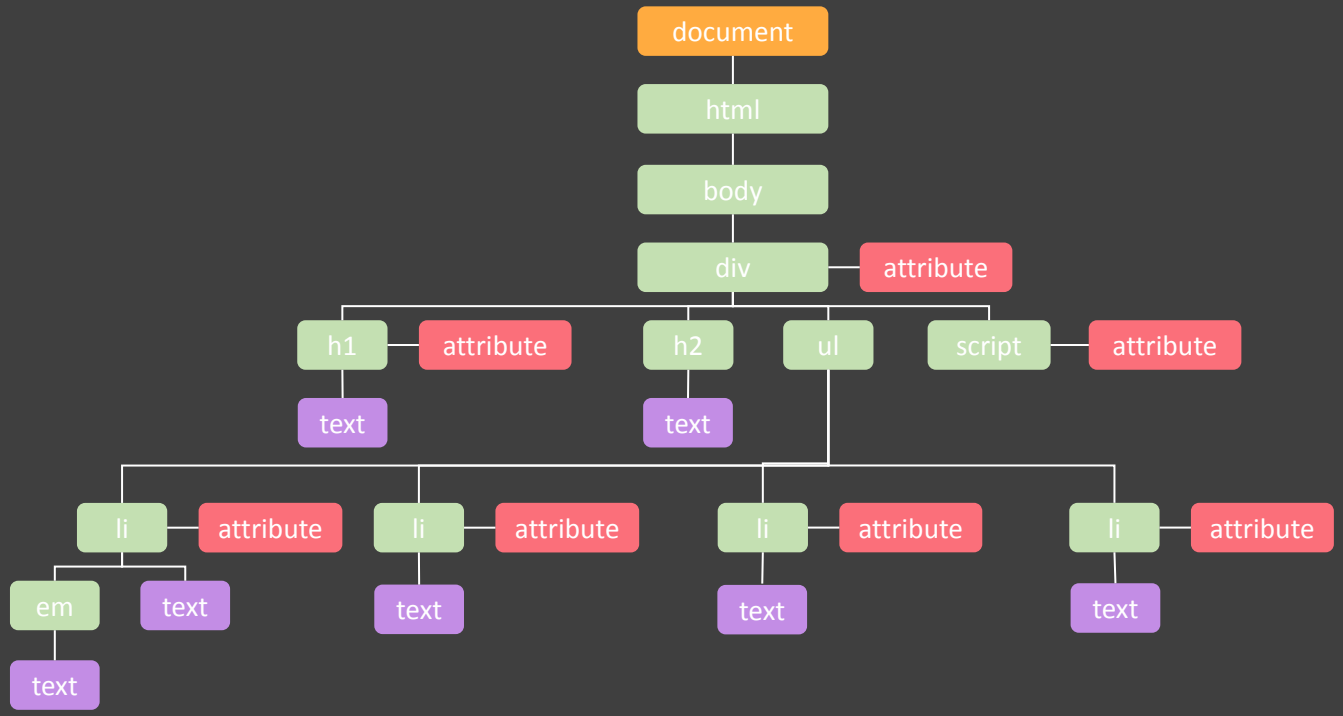
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Simple List</title>
</head>
<body>
    <div id="page">
        <h1 id="header">List</h1>
        <h2>Buy groceries</h2>
    </div>
    <ul>
        <li id="one" class="hot"><em>fresh</em> grapes</li>
        <li id="two" class="hot">peanuts</li>
        <li id="three" class="hot">strawberry jam</li>
        <li id="four">olive oil</li>
    </ul>
    <script src="js/dom.js"></script>
</body>
</html>
```

SAMPLE HTML PAGE & DOM

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Simple List</title>
</head>
<body>
    <div id="page">
        <h1 id="header">List</h1>
        <h2>Buy groceries</h2>
    </div>
    <ul>
        <li id="one" class="hot"><em>fresh</em> grapes</li>
        <li id="two" class="hot">peanuts</li>
        <li id="three" class="hot">strawberry jam</li>
        <li id="four">olive oil</li>
    </ul>
    <script src="js/dom.js"></script>
</body>
</html>
```



DOM TREE



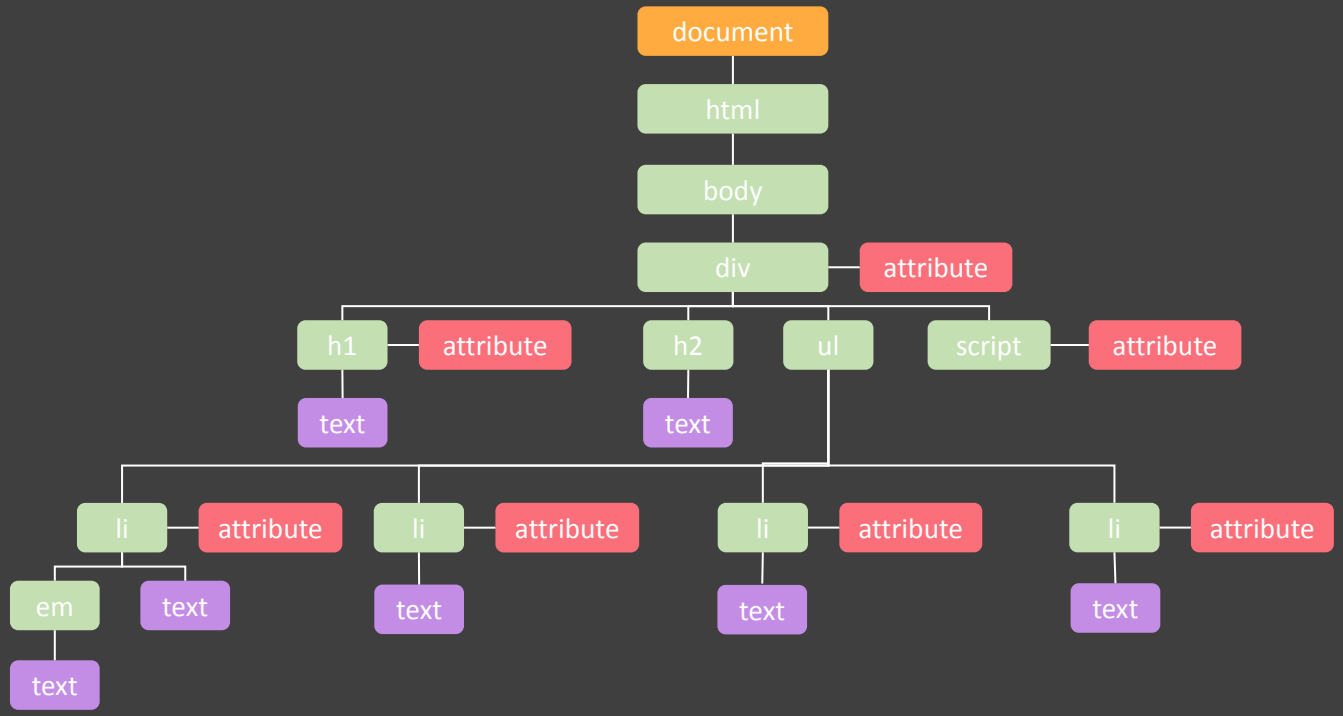
- DOCUMENT NODE
- ELEMENT NODE
- ATTRIBUTE NODE
- TEXT NODE

DOCUMENT NODE

Every element, attribute, text in HTML is represented by its own DOM node.

At the top, resides the document node. It represents the entire page document. When you access any node, you navigate via the document node.

DOM TREE



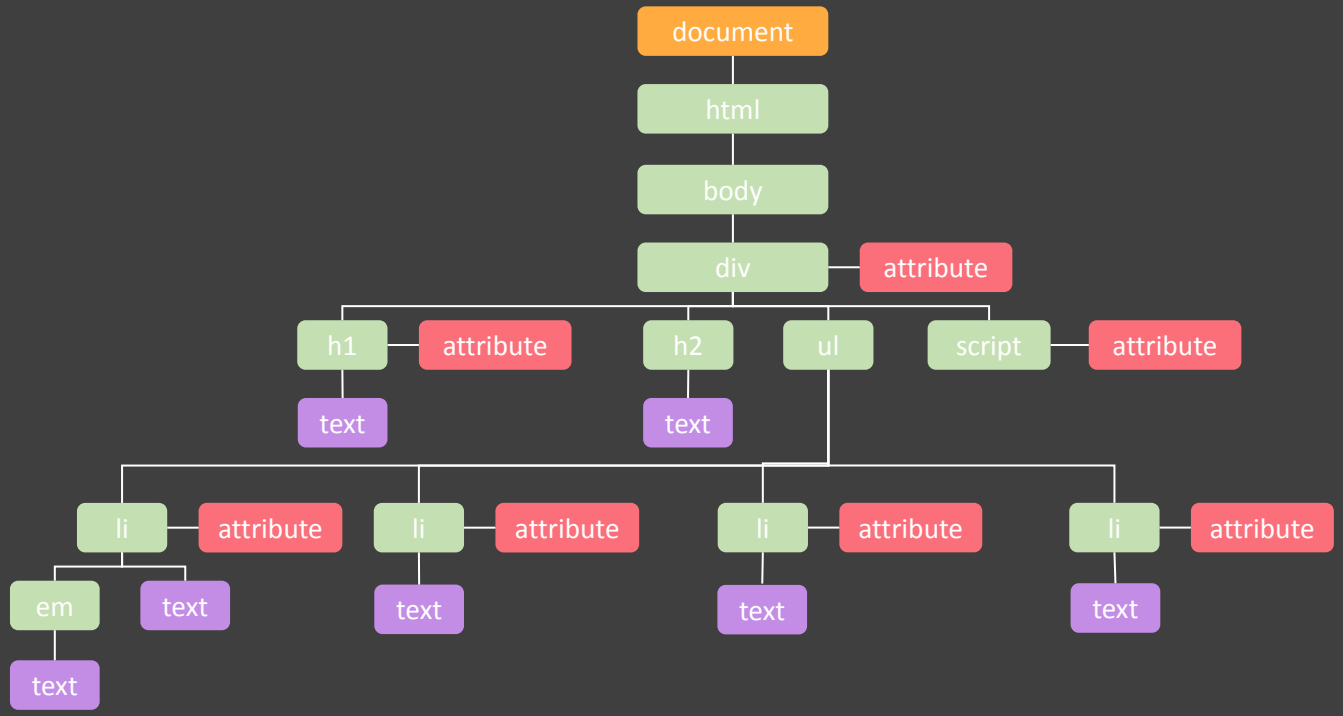
- DOCUMENT NODE
- ELEMENT NODE
- ATTRIBUTE NODE
- TEXT NODE

ELEMENT NODES

HTML elements describe the structure of an HTML document.

To access the DOM tree, you start by looking for element nodes (HTML tags). Once you locate the node, you can access its attribute or text nodes.

DOM TREE



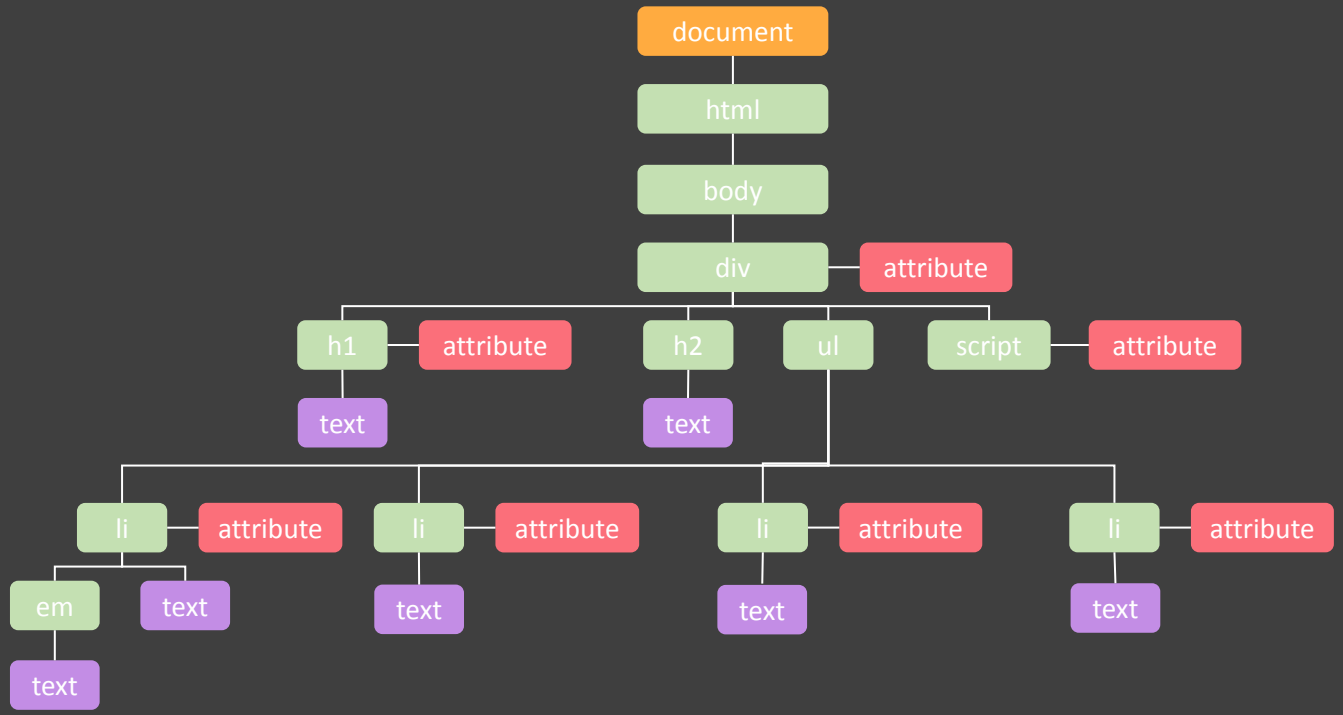
- DOCUMENT NODE
- ELEMENT NODE
- ATTRIBUTE NODE
- TEXT NODE

ATTRIBUTE NODES

The opening tags of HTML elements can carry attributes eg. class="hot", id="one"

Attribute nodes are not the child of the element but are part of it.
We can use JavaScript to modify the attribute

DOM TREE



- DOCUMENT NODE
- ELEMENT NODE
- ATTRIBUTE NODE
- TEXT NODE

TEXT NODES

Once you can access an element node, you can reach the text within and modify it.

Text nodes cannot have children.

Working with the DOM

Document Object Model

STEPS

1. Locate the node that represents the element you want to work with

2. Use its text content, child elements, and attributes

DOM Manipulation Basics

```html
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width">
   <title>repl.it</title>
   <link href="style.css" rel="stylesheet" type="text/css" />
 </head>
 <body>
   <h1 id="title">Hello World</h1>
   <input id="my-button" type="button" value="Click me">
 </body>
</html>
```

#title

Hello World

Click me

#my-button

Evaluating the Statement

```
let header = document.getElementById('title');
header.innerHTML = "Goodbye World"
```

Evaluating the statement 2

```
let header = document.getElementById('title');
⇒ DOM Object <#title>
header.innerHTML = "Goodbye World"
⇒ DOM Object <#tite>.innerHTML = "Goodbye World"
```

Structure of Manipulation

```
let header = document.getElementById('title');
header.innerHTML = "Goodbye World"
```

Step 1 - **SELECT** the element

Step 2 - **MODIFY** the element

```
document.getElementById('title').innerHTML = "Goodbye World"
```

What's Going On

```
document.getElementById('title').innerHTML = "Goodbye World"
```

SELECT THE ELEMENT

MANIPULATION

1. ACCESSING ELEMENTS

`getElementById();`

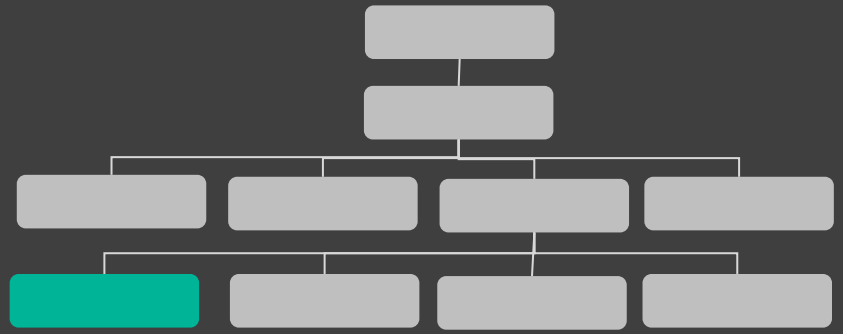
Uses the value of an element's id attribute

`querySelector();`

Uses a CSS selector, and returns the **first** matching element. Classes or Ids can be used

`Traversal methods`

SELECT INDIVIDUAL NODE ELEMENT



https://replit.com/@immalcolm/Simple-DOM

1. ACCESSING ELEMENTS

`getElementsByClassName();`

Selects all elements that have a specific value for their class attribute

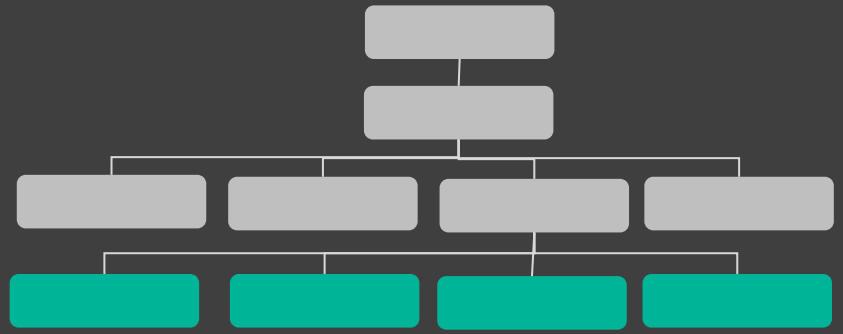
`getElementsByTagName();`

Selects all elements that have the specified tag name

`querySelectorAll()`

Uses a CSS selector to select all matching elements

SELECT MULTIPLE ELEMENTS (NODE LISTS)



1. ACCESSING ELEMENTS (TRAVERSAL)

`parentNode`

Selects the parent of the current node

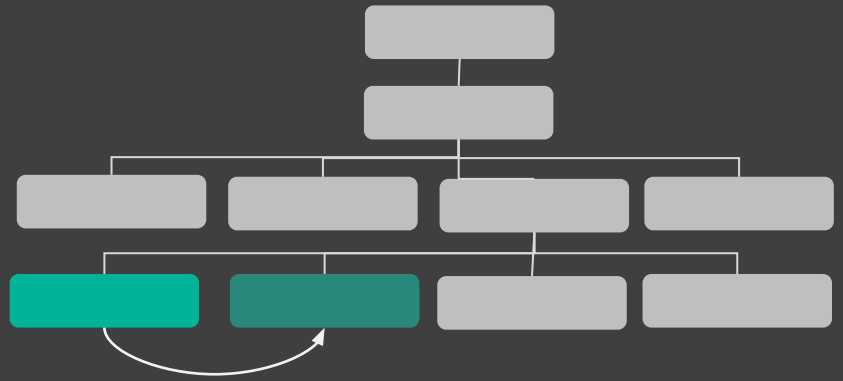
`previousSibling/nextSibling`

Selects the previous or next sibling from the DOM tree

`firstChild/lastChild`

Selects the first or last child of the current element

TRAVERSING BETWEEN ELEMENT NODES



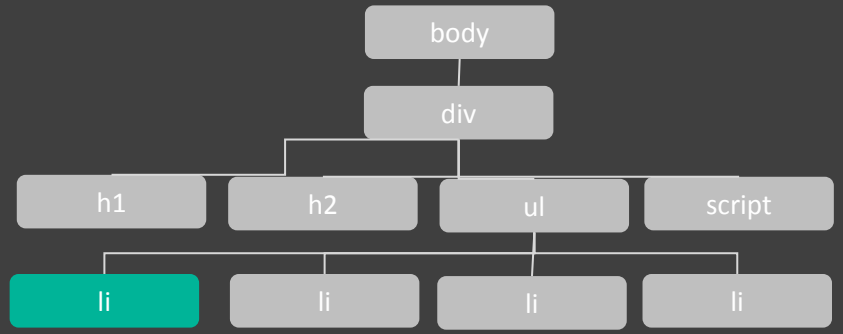
1. ACCESSING ELEMENTS

When a script selects an element to modify, the interpreter finds the element(s) in the DOM tree

The following query searches the DOM tree for an element whose **id** attribute has a value of **one**

Once the node is found, you can work with that node, its parent or children

SELECT INDIVIDUAL NODE ELEMENT



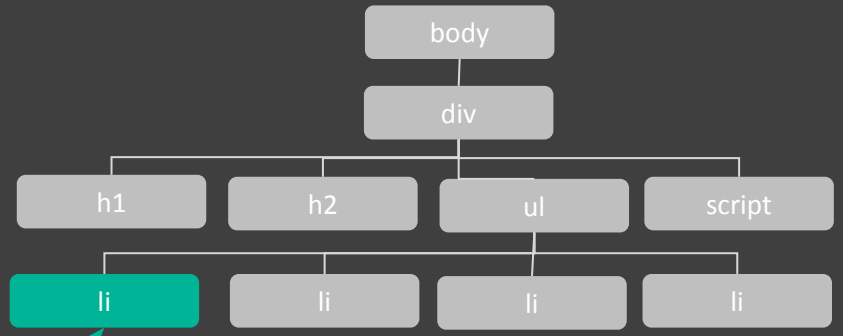

```
getElementById("one");
```

https://replit.com/@immalcolm/Simple-DOM

1. ACCESSING ELEMENTS

If you are working with the element more than once, it is best you store it in a variable.

```
var itemOne = getElementById("one");
```



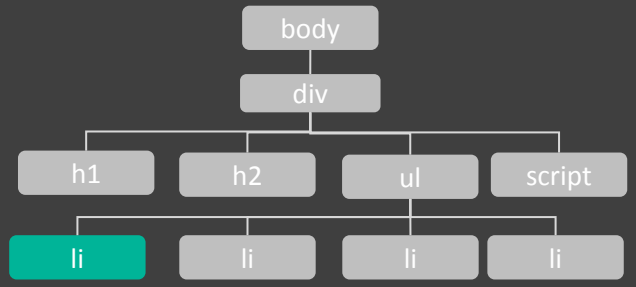
Saves repeated search queries. This is known as storing a **reference** of the object

SELECT INDIVIDUAL NODE ELEMENT

1. ACCESSING ELEMENTS

`getElementById('id');`

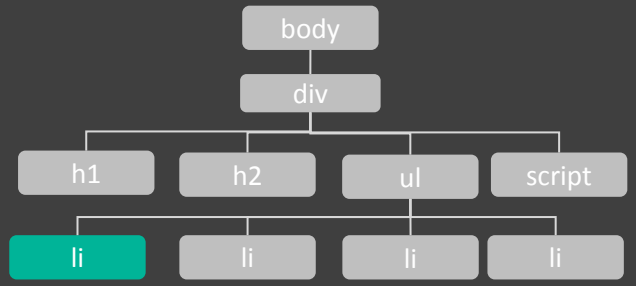
Selects an individual element given the value of its id.
The HTML must have an id attribute.



`getElementById('one');`

`querySelector('css selector');`

Selects an individual element given the value of its id.
The HTML must have an id attribute.



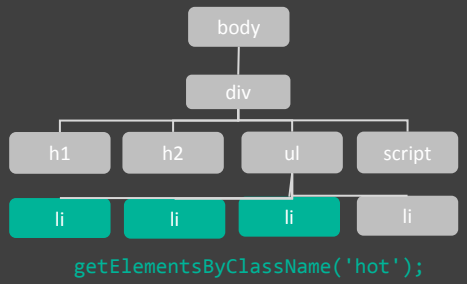

```
querySelector('li.hot');
querySelector('#one');
```

SELECT **MULTIPLE** ELEMENTS (NODE LISTS)

1. ACCESSING ELEMENTS

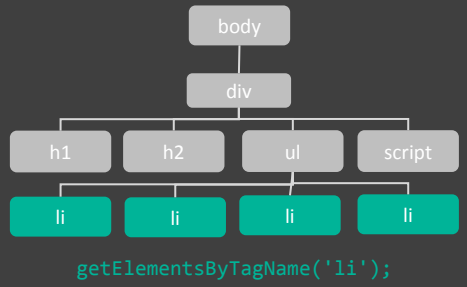
`getElementsByClassName('class');`

Selects one or more elements given the value of their class attribute.



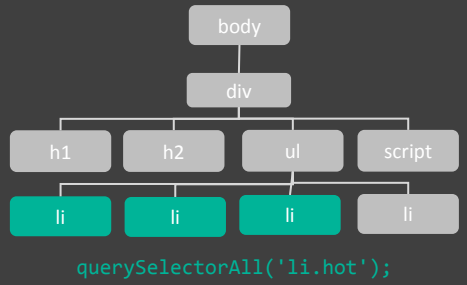
`getElementsByTagName('tagName');`

Selects all elements on the page with the specified tag name



`querySelectorAll('css selector');`

Uses CSS selector syntax to select one or more elements and returns all that matches



document refers to the **document** object. You would always have to access individual elements via the **document** object

OBJECT

The **getElementById()** method indicates you want to search an element based on the **id** attribute

METHOD (selector)

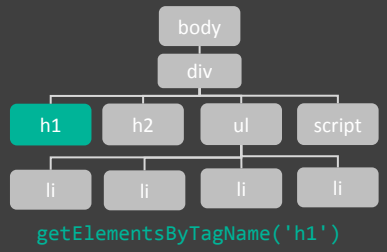
```
document.getElementById('one');
```

MEMBER OPERATOR

The dot notation indicates that the method (on the right) is being applied to the node on the left of the period

PARAMETER

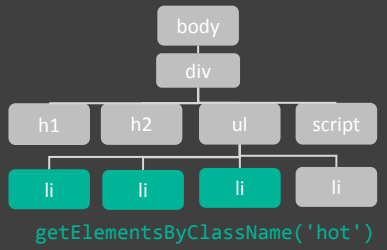
The method needs to know the value of the **id** attribute on the element you want.



`getElementsByTagName('h1');`

Even though the query returns only one element, the method returns a node list

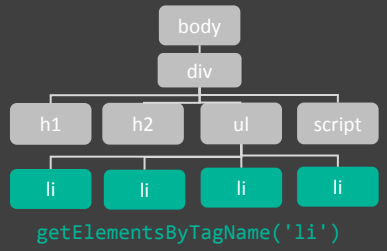
INDEX	NUMBER & ELEMENT
0	<h1>



`getElementsByClassName('hot');`

Node list contains only three of the <li> elements, because we are searching by class attribute

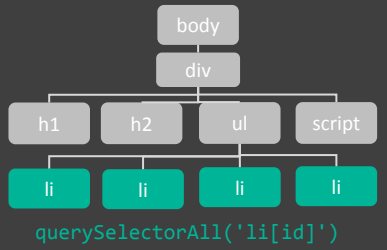
INDEX	NUMBER & ELEMENT
0	<li id="one" class="hot">
1	<li id="two" class="hot">
2	<id id="three" class="hot">



`getElementsByTagName('li');`

This method returns four elements, one for each <li> element. Following the order of their HTML appearance

INDEX	NUMBER & ELEMENT
0	<li id="one" class="hot">
1	<li id="two" class="hot">
2	<id id="three" class="hot">
3	<li id="four>



`querySelectorAll('li[id]');`

The query returns four elements, one for each <li> attribute that contains an id attribute

INDEX	NUMBER & ELEMENT
0	<li id="one" class="hot">>
1	<li id="two" class="hot">>
2	<id id="three" class="hot">>
3	<li id="four>

https://replit.com/@immalcolm/Simple-DOM

item() method

Nodelists have a method called items()
Returns the individual node from the NodeList.
Specify the index number as the parameter

```
let elements =
document.getElementsByClassName('hot');
if (elements.length >= 1){
    let firstItem = elements.item(0);
}
```

ARRAY syntax

Access individual nodes using a square bracket syntax like an array. Specify the index number as the parameter

```
let elements =
document.getElementsByClassName('hot');
if (elements.length >= 1){
    let firstItem = elements[0];
}
```

1

Create a **NodeList** containing elements that have a **class** attribute whose value is hot, and store into the variable elements

2

If that number **is greater than or equal to 1,** meaning that there's at least one element in the NodeList then run the code within

3

Get the first element from the NodeList.
0 == first
(because index numbers in JS starts from zero)

Refers to the element node that has been retrieved

OBJECT

The **className property allows us to set or get** the class attribute of the node

METHOD

```
<element>.className;
```

MEMBER OPERATOR

The dot notation indicates that the method (on the right) is being applied to the node on the left of the period

```
let el =
document.getElementById('one');

//retrieve the class involved
console.log(el.className);

//set a new class
el.className = 'cool';
```

Additional Reading: https://www.w3schools.com/jsref/prop_html_classname.asp

Refers to the element node that has been retrieved

OBJECT

The innerHTML property allows retrieval of content of an element or modification of new content. **HTML markup is NOT allowed**

PROPERTY

```
<element>.textContent;
```

GET CONTENT

```
var nodeContent =
document.getElementById('one').textContent;
```

SET CONTENT

```
document.getElementById('one').textContent=
"fresh";
```

Refers to the element node that has been retrieved

The innerHTML property allows retrieval of content of an element or modification of new content. **HTML markup is allowed**

OBJECT

PROPERTY

```
<element>.innerHTML;
```

GET CONTENT

```
let nodeContent =
document.getElementById('one').innerHTML;
```

SET CONTENT

```
document.getElementById('one').innerHTML =
"<b>fresh</b>;
```

#DOM
#NODES

Live NodeList

When your script updates the page, the NodeList is updated at the same time. The methods beginning **getElementsBy ..** Return live NodeLists.

They are typically faster to generate than static NodeLists

Usage: When you are dependent on live changes

```
getElementById()
getElementsByClassName()
getElementsByTagName
```

Static NodeList

When your script updates the page, the NodeList is not updated to reflect the changes made by the script

querySelector… (which uses CSS selector syntax) return static NodeLists. They reflect the document when the query was made. If the script changes the content of the page, the NodeList is not updated to reflect those changes.

Usage: when you are not dependent of node changes

```
querySelector()
querySelectorAll()
```

https://replit.com/@immalcolm/nodelist-compare-static-vs-live

ℹ️ *It's good to keep this distinction in mind when you choose how to iterate over the items in the NodeList, and whether you should cache the list's length.*

When to Use each Selector

SPECIFIC: SELECT ONE

- getElementById
- getElementByName

SPECIFIC: SELECT MANY

- getElementsByClassName
- getElementsByTagName
- getElementsByName

GENERAL: SELECT ONE

- querySelector

GENERAL: SELECT MANY

- querySelectorAll

Event Listener

STORY

When you browse the web, the browser register different types of events.

It's the browser way of saying "Hey, this just happened" & your script responds to these events

Scripts often respond to these events by updating the contents of the webpage (via the DOM)

THE CYCLE

INTERACTIONS CREATE EVENTS

Events occur when users click or tap on a link, hover or swipe over an element, type on the keyboard, resize the window or when the page requested has loaded

EVENTS TRIGGER CODE

When an event occurs, or fires, it can be used to trigger a particular function. Different code can be triggered when users interact with different parts of the page.

CODE RESPONDS TO USERS

Events can trigger the changes to the DOM. This is how a web page reacts to users

Intro to Event Driven

MOST OF THE JAVASCRIPT IS IMPERATIVE

That is, the instructions are executed **immediately** when JavaScript reaches them, as in Python.

Event driven programming is DECLARATIVE

It likes setting a standing instruction.

<u>It is vital to differentiate the two of them as both exists in JavaScript *in the same program*!</u>

Imperative programming is a paradigm describing **HOW** the program should do something by explicitly specifying each instruction (or statement) step by step, which mutate the program's state.

Declarative programming is a paradigm describing **WHAT** the program does, without explicitly specifying its control flow.

Event Listener

What event to detect

```
document.getElementById('my-button').addEventListener('click', function(){
  let header = document.getElementById('title');
  header.innerHTML = "Goodbye World"
})
```

Hello World

Click me

What function to run when the event happens to the element

IMPORTANT!!! *AddEventListener is* ***declarative***, not imperative

UI Events

Occur when a user interacts with the browser's user interface (UI) rather than the webpage

EVENT	DESCRIPTION
load	Web page has finished loading
unload	Web page is unloading (usually because a new page was requested
error	Browser encounters a JavaScript error or an asset doesn't exist
resize	Browser window has been resized
scroll	User has scrolled up or down the page

https://javascript.info/event-details
https://www.w3schools.com/jsref/obj_events.asp

KEYBOARD EVENTS

Occur when a user interacts with the keyboard (aka input events)

EVENT	DESCRIPTION
keydown	User first presses a key (repeats while key is depressed)
keyup	User releases a key
keypress	Character is being inserted (repeats while key is depressed

MOUSE EVENTS

Occur when a user interacts with a mouse, trackpad, or touchscreen

EVENT	DESCRIPTION
click	User presses and releases a button over the same element
dblclick	User presses and releases a button twice over the same element
mousedown	User presses a mouse button while over an element
mouseup	User releases a mouse button while over an element
mousemove	User moves the mouse (not on a touchscreen)
mouseover	User moves the mouse over an element (not on a touchscreen)
mouseout	User moves the mouseoff an element (not on a touchscreen)

FOCUS EVENTS

Occur when an element (e.g, a link or form field) gains or loses focus

EVENT	DESCRIPTION
focus / focusin	Element gains focus
blur / focusout	Element loses focus

FORM EVENTS

Occur when a user interacts with a form element

EVENT	DESCRIPTION
input	Value in any <input> or <textarea> element has changed or any element with the contenteditable attribute
change	Value in select box, checkbox, or radio button changes
submit	User submits a form (using a button or a key)
reset	User clicks on a form's reset button (rarely used in today's context) UX reasons
cut	User cuts content from a form field
copy	User copies content from a form field
paste	User pastes content into a form field
select	User selects some text in a form field

1

Select the **element** node(s) you want the script to respond to

Eg. If you want to trigger a function when a user clicks on a specific link, you need to get the DOM node for that element (via DOM query)

2

Indicate which **event** on the selected node(s) will trigger the response.

Binding an event to a DOM node

3

State the **code** you want to run when the event occurs.

When the event occurs, on a specified element, it will trigger a function. This may be a named or anonymous function

SCENARIO

Event handling used to provide feedback when users fill in a registration form. It will show an error message if the username is too short.

1 SELECT ELEMENT

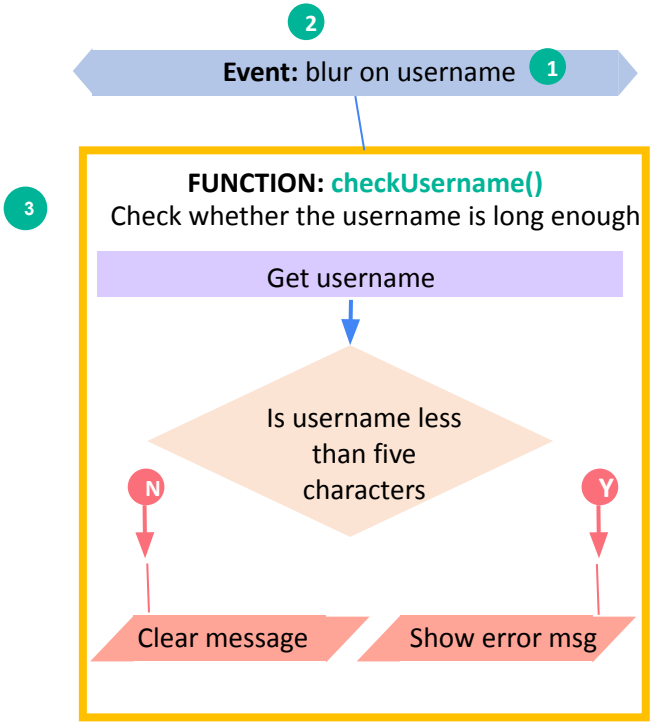
The element that users are interacting with is the text input where they enter the username

2 SPECIFY EVENT

When users move out of the text input, it loses focus, and the blur event first on this element

3 CALL CODE

When the blur event fires on the username input, it will trigger a function called **checkUsername()**. This function checks if the username is less than 5 characters



https://replit.com/@immalcolm/sample-username-check

Event Listeners

EVENT LISTENER WITH NO PARAMETERS (VANILLA JS)

Adds an event listener to the DOM element node(s)

METHOD

```
element.addEventListener('event', functionName, [Boolean]);
```

ELEMENT
DOM element node to target

EVENT
Event to bind node(s) to in quote marks

CODE
Name of function to call

EVENT FLOW
Indicates something called capture, and is usually set to false

```
function checkUsername(){
   //code to check the length of username
}
var el = document.getElementById('username');
el.addEventListener('blur', checkUsername, false);
```

The function is called by the event listener but parentheses are omitted

Event name in quotation marks

https://replit.com/@immalcolm/sample-username-check

Refer to activity/js/event-listener.js

EVENT LISTENER WITH PARAMETERS (VANILLA JS)

Declarative -- *standing instruction. ONLY WHEN the button is pressed, then the function happens*

Event name

Start of anonymous function

```
el.addEventListener('blur', function(){
  checkUsername(5);
}, false);
```

Imperative. It only happens when the program reaches this point.

The **anonymous function** is used as the second argument. It "wraps around" the named function.

End of statement

End of addEventListener() method

Event flow boolean

End of anonymous function

Because you cannot have parentheses after the function names in event handlers or listeners, passing arguments requires a workaround with anonymous function

https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener
https://www.w3schools.com/js/js_function_definition.asp
https://stackoverflow.com/questions/5657292/why-is-false-used-after-this-simple-addeventlistener-function

Event Objects

EVENT OBJECT

When an event occurs, the event object tells information about the event, and the element it happened upon.

Every time an event fires, the event object contains helpful data about the event

- *Which element the event happened on*
- *Which key was pressed for a keypress event*
- *What part of the viewport the user clicked for a click event*

The event object is passed to any function that is the event handler or listener

EVENT LISTENER WITH NO PARAMETERS (A)

```
function checkUsername(event){
  var target = event.target; //get target of event
  //do something
}

var el = document.getElementById('username');
el.addEventListener('blur', checkUsername, false);
```

EVENT LISTENER WITH PARAMETERS (B)

```
function checkUsername(event, minLength){
  var target = event.target; //get target of event
  //do something
}

var el = document.getElementById('username');
el.addEventListener('blur', function(event){
  checkUsername(event, 5);
}, false);
```

https://replit.com/@immalcolm/sample-username-check

Refer to activity/js/event-listener-with-event-object.js
https://replit.com/@immalcolm/activity-materials

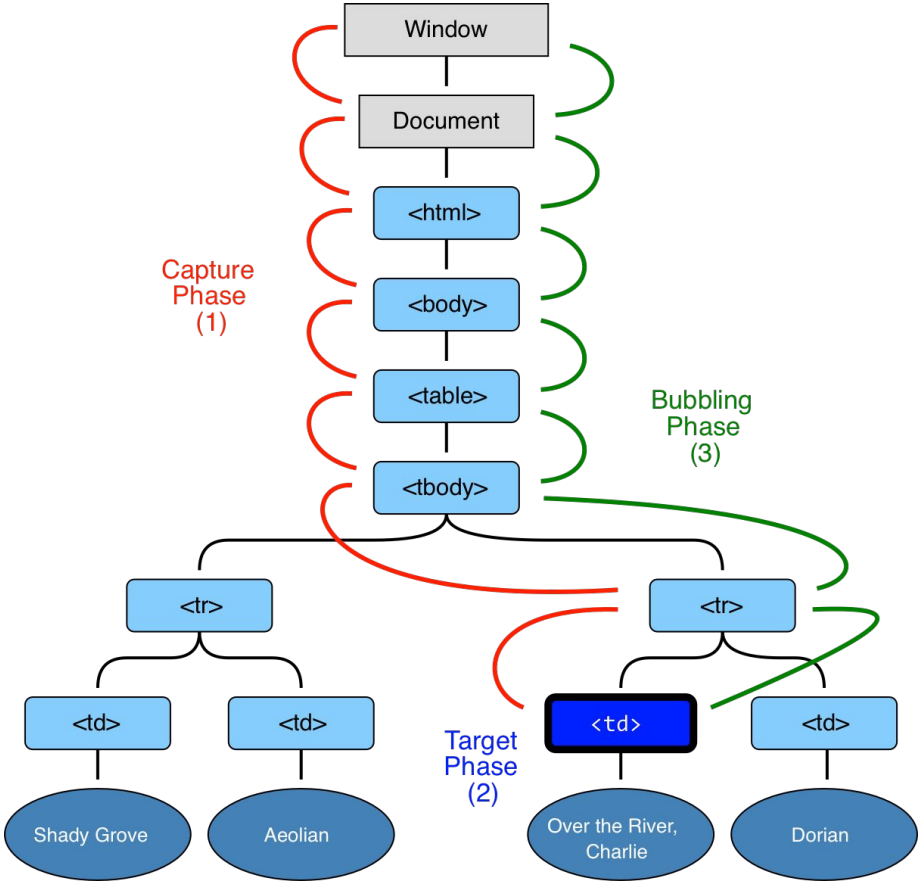
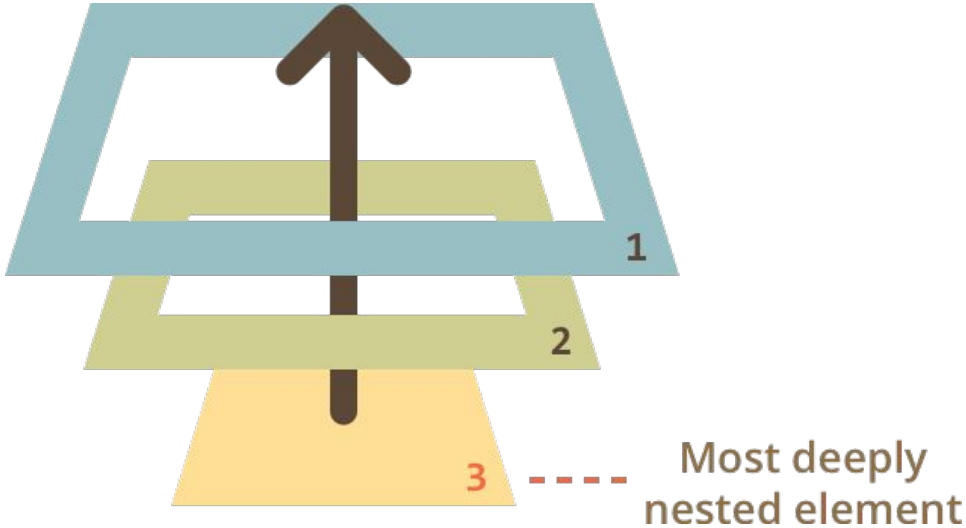


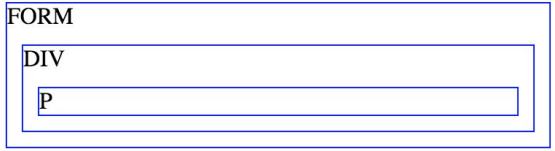
Image Source: https://www.w3.org/TR/DOM-Level-3-Events/
Reading: https://javascript.info/bubbling-and-capturing

EVENT BUBBLING




```
<style>
  body * {
    margin: 10px;
    border: 1px solid blue;
  }
</style>

<form onclick="alert('form')">FORM
  <div onclick="alert('div')">DIV
    <p onclick="alert('p')">P</p>
  </div>
</form>
```



A click on the inner <p> first runs onclick.

1. On that <p>.
2. Then on the outer .
3. Then on the outer <form>.
4. And so on upwards till the document object.

Example: https://javascript.info/article/bubbling-and-capturing/bubble-target/

Try it out https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_event_stoppropagation

See sample code event-bubbling.html

CHANGING DEFAULT BEHAVIOR

preventDefault()

Some events such as clicking on a link or submitting a form, brings the user to another page.

To prevent the default behavior of such element (e.g to keep the user on the same page to perform validation) you can use the event object's preventDefault() method.

Prevents the default action of the event from triggering. Does not stop the event propagation to parent DOM elements.

event.preventDefault()

stopPropagation()

Once you have handled an event using one element, you may want to stop that event from bubbling up to its ancestor elements.

To stop the event bubbling up, you can use the event's object's stopPropagation method.

Prevents the default action of the event from propagating to parent DOM elements.

Try it out
https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_event_stoppropagation

event.stopPropagation()

Recommended Reading Reference: https://javascript.info/bubbling-and-capturing
https://www.w3schools.com/jsref/event_stoppropagation.asp

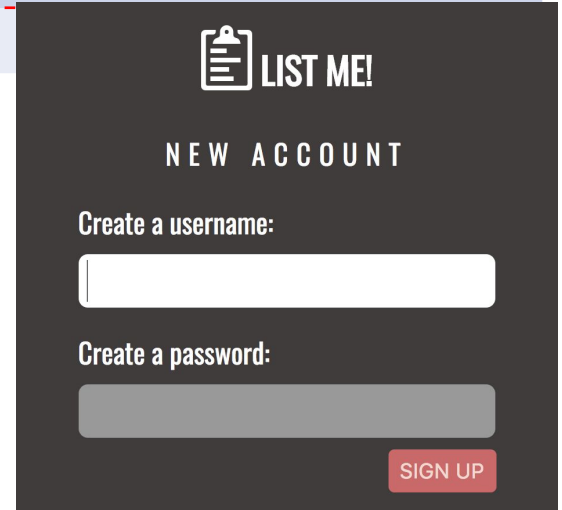
UI Events

Occur when a user interacts with the browser's user interface (UI) rather than the webpage

EVENT	DESCRIPTION
load	Web page has finished loading
unload	Web page is unloading (usually because a new page was requested
error	Browser encounters a JavaScript error or an asset doesn't exist
resize	Browser window has been resized
scroll	User has scrolled up or down the page

```
function setup() {                                  // Declare function
  var textInput;                                    // Create variable
  textInput = document.getElementById('username');  // Get username input
  textInput.focus();                                // Give username focus
}

window.addEventListener('load', setup, false); // When page loaded call setup()
```



Refer to sample activity code load.js load.html

https://replit.com/@immalcolm/jsevent-activity-materials
https://jsevent-activity-materials.immalcolm.repl.co/load.html

FOCUS EVENTS

Occur when an element (e.g, a link or form field) gains or loses focus
You want to show tips or feedback to users as they interact with the page elements.
You need to trigger form validation as a user moves from one control to the next

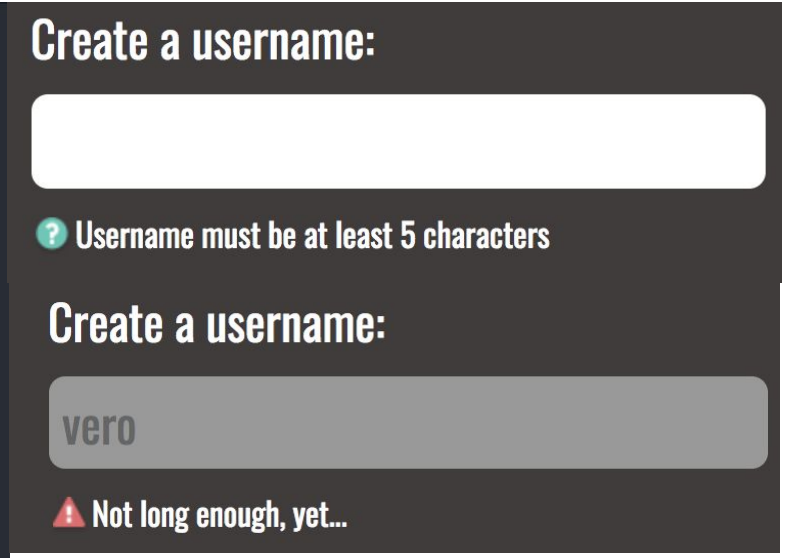
EVENT	DESCRIPTION
focus / focusin	Element gains focus
blur / focusout	Element loses focus

```
function checkUsername() {                          // Declare function
  var username = el.value;                          // Store username in variable
  if (username.length < 5) {                       // If username < 5 characters
    elMsg.className = 'warning';                   // Change class on message
    elMsg.textContent = 'Not long enough, yet...';// Update message
  } else {                                         // Otherwise
    elMsg.textContent = '';                        // Clear the message
  }
}

function tipUsername() {                            // Declare function
  elMsg.className = 'tip';                         // Change class for message
  elMsg.innerHTML = 'Username must be at least 5 characters'; // Add message
}

var el = document.getElementById('username');      // Username input
var elMsg = document.getElementById('feedback');   // Element to hold message

// When the username input gains / loses focus call functions above:
el.addEventListener('focus', tipUsername, false); // focus call tipUsername()
el.addEventListener('blur', checkUsername, false);// blur call checkUsername()
```



Refer to sample activity code focus-blur.html
https://jsevent-activity-materials.immalcolm.repl.co/focus-blur.html

MOUSE EVENTS

Occur when a user interacts with a mouse, trackpad, or touchscreen

EVENT	DESCRIPTION
click	User presses and releases a button over the same element
dblclick	User presses and releases a button twice over the same element
mousedown	User presses a mouse button while over an element
mouseup	User releases a mouse button while over an element
mousemove	User moves the mouse (not on a touchscreen)
mouseover	User moves the mouse over an element (not on a touchscreen)
mouseout	User moves the mouseoff an element (not on a touchscreen)

https://replit.com/@immalcolm/click-eventhandler

Events with Forms

Y VALIDATE?

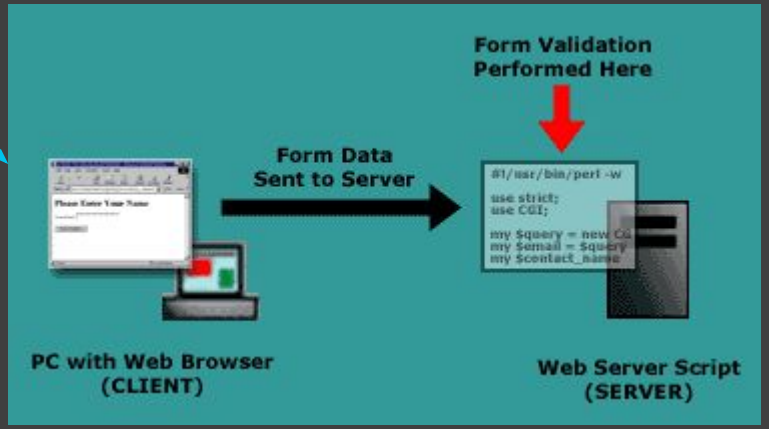
Prevent user from enter invalid inputs

We have *server-side validation* and *client-side validation*

- Client = browser, mobile phone browser etc.

Client vs. Server Validation

Doing validation with JavaScript happens here



Takes place before data is sent to server

Server will usually validate again

Why Validate on Client?

If we're going to validate again on server, why perform validation on client-side?

Instant feedback

Save load on server

Focus vs Blur

Focus is called when a HTML element has the focus - i.e. has the mouse cursor

Blur is called when a HTML element lose the focus

Forms are used to get input from the users
Input elements must be in the form element

```
<form>
  <input type="text"></input>
  <!– insert form elements code-->
</form>
```

TEXTBOXES

```
<input type="text" name="txtUsername" id="txtUsername">
```

Email address

Enter email

```
document.getElementById("txtUserName").value;
//allows us to retrieve the value of the input box
```

RADIO BUTTONS

Vanilla Chocolate Strawberry

Shares the same name

```
<input type="radio" name="rdoFlavour" value="vanilla">Vanilla
<input type="radio" name="rdoFlavour" value="chocolate">Chocolate
<input type="radio" name="rdoFlavour" value="strawberry" disabled>Strawberry
```

Assign value to the element. This is retrieved via JS later

https://replit.com/@immalcolm/sample-event-checkbox-radio

```
let rdoValue =
document.querySelector('input[name="rdoFlavor"]:checked').value;
```

CHECK BOX

Shares the same name

```
<input type="checkbox" name="chkStyle" value="cone">Cone
<input type="checkbox" name="chkStyle" value="wafer">Wafer
<input type="checkbox" name="chkStyle" value="bread" disabled>Bread
```

Assign value to the element. This is retrieved via JS later

☐ Wafer ☐ Cone ☐ Bread

https://replit.com/@immalcolm/sample-event-checkbox-radio

DROPDOWN MENU

Yishun

```
<select name="locations" id="locations">
  <option value="north-yishun">Yishun</option>
  <option value="east-changi">Changi</option>
  <option value="west-clementi">Clementi</option>
</select>

//retrieve selected option
let locations = document.getElementById('locations').value;
```

https://www.w3schools.com/jsref/prop_select_value.asp
https://replit.com/@immalcolm/sample-event-select

DROPDOWN MENU

```
<select multiple name="locations">
  <option value="north-yishun">Yishun</option>
  <option value="east-changi">Changi</option>
  <option value="west-clementi">Clementi</option>
</select>
```

Multiple Select

https://replit.com/@immalcolm/sample-event-select

SELECT & RETRIEVE FORM ELEMENT

By assigning a id or a class (like normal HTML)
By selecting by name:

```
document.forms[<form name>][<element name>].value;
```

JS

```
document.forms['myfrm']['username'].value; //based on name attribute
//document.getElementById('username').value; //alternative
```

HTML

```
<form name='myfrm'>
  <label for='username'>
  <input type='textbox' name='username' id='username'>
</form>
```

Example - KeyUp

```
<element>.addEventListener('keyup',function(e) {
    //do something
},false)
```

Key Takeaway?

Practise & Practise

We simply don’t trust the user. Always validate inputs and start early. It’s alright to be paranoid then loosen up.

Learning Resources

Visual Learning CSS & JS: https://www.codeanalogies.com/

Learn JS with MDN: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps