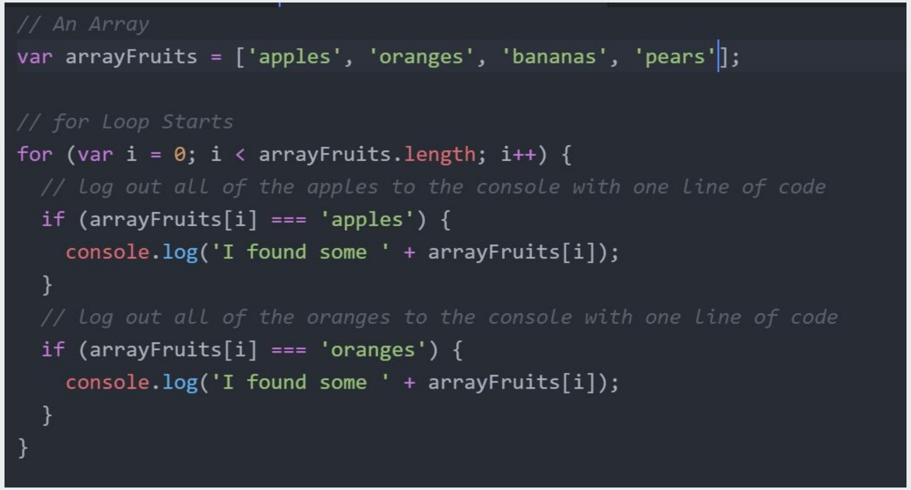
Foundation Coding - Week 3: Basics

Loops

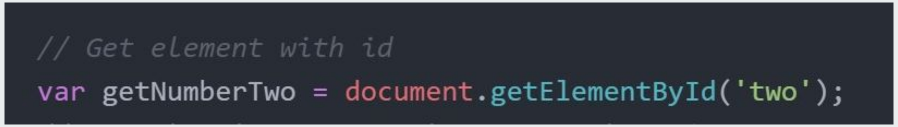
- Loops are used to iterate and perform code 1 or more times
- Loop over an array
- *i* is used as the index reference (number)
- *A for loop* is a well known traditional method of performing loops
- There are other kinds of loops (*while, do while , for in*)
- A traditional for loop will perform most tasks needed from a loop



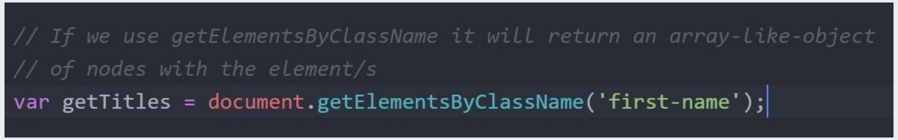
```
// An Array
var arrayFruits = ['apples', 'oranges', 'bananas', 'pears'];

// for Loop Starts
for (var i = 0; i < arrayFruits.length; i++) {
  // log out all of the apples to the console with one line of code
  if (arrayFruits[i] === 'apples') {
    console.log('I found some ' + arrayFruits[i]);
  }
  // log out all of the oranges to the console with one line of code
  if (arrayFruits[i] === 'oranges') {
    console.log('I found some ' + arrayFruits[i]);
  }
}
```

DOM Queries

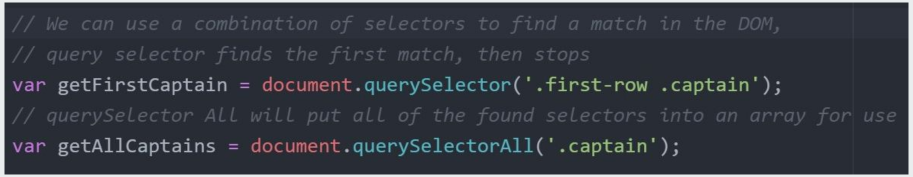
- When using vanilla .js there are various ways of accessing DOM elements
- The four most popular are by class, id, querySelector and querySelectorAll
- *getElementById()* and *querySelector()* will retrieve a single match
- *getElementsByClassName()* and *querySelectorAll()* will return an array of element/s



```
// Get element with id
var getNumberTwo = document.getElementById('two');
```



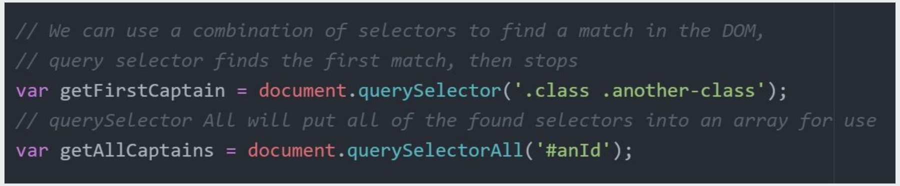
```
// If we use getElementsByClassName it will return an array-like-object
// of nodes with the element/s
var getTitles = document.getElementsByClassName('first-name');
```



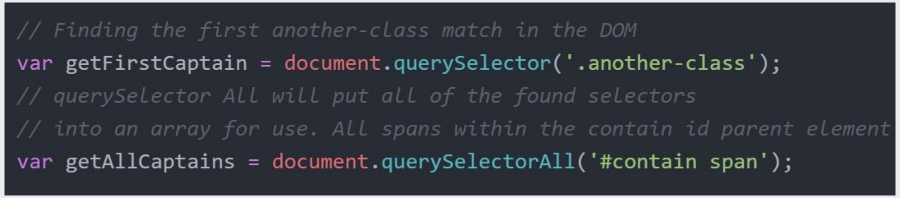
```
// We can use a combination of selectors to find a match in the DOM,
// query selector finds the first match, then stops
var getFirstCaptain = document.querySelector('.first-row .captain');
// querySelector All will put all of the found selectors into an array for use
var getAllCaptains = document.querySelectorAll('.captain');
```

DOM Queries

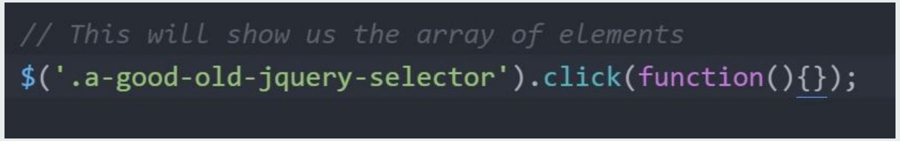
- *querySelector()* and *querySelectorAll()* can use elements, classes and ids as selectors.
- If a class or id is used you must add a dot *.class* or hash *#id* to the selector in the .js query
- querySelector/s are a good option for getting elements. Choose up to 2 DOM queries and stick with them.
-...You can always use jQuery if needed as well



```
// We can use a combination of selectors to find a match in the DOM,
// query selector finds the first match, then stops
var getFirstCaptain = document.querySelector('.class .another-class');
// querySelector All will put all of the found selectors into an array for use
var getAllCaptains = document.querySelectorAll('#anId');
```



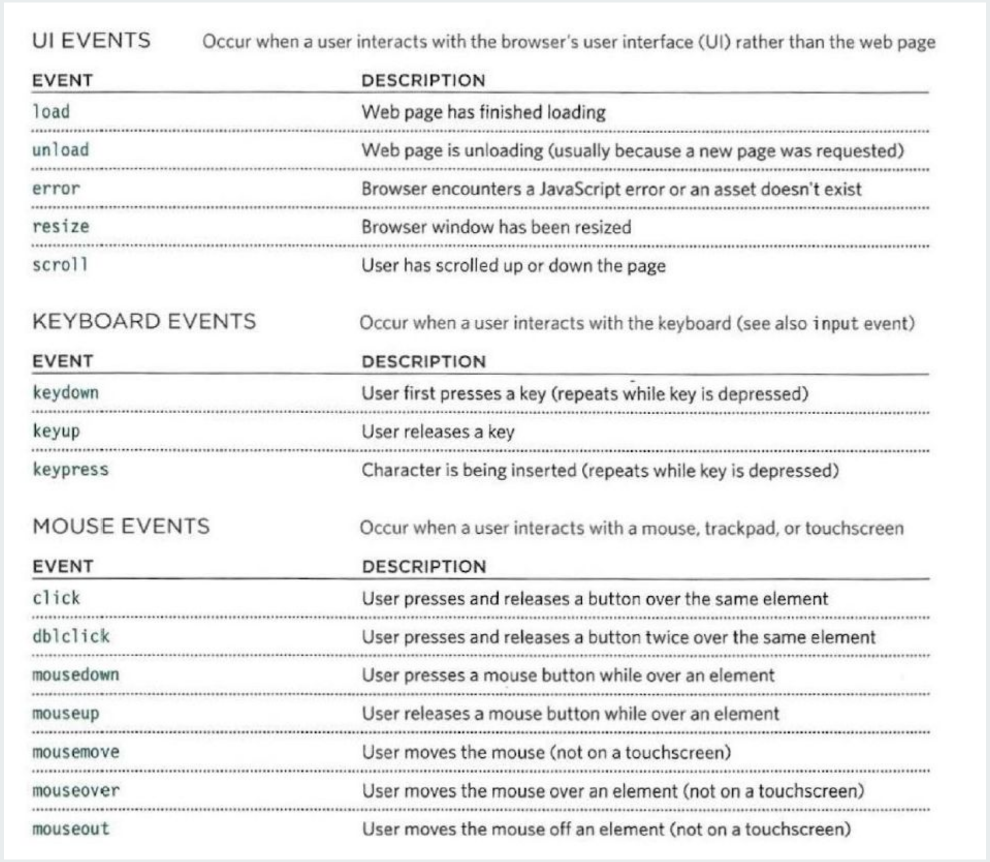
```
// Finding the first another-class match in the DOM
var getFirstCaptain = document.querySelector('.another-class');
// querySelector All will put all of the found selectors
// into an array for use. All spans within the contain id parent element
var getAllCaptains = document.querySelectorAll('#contain span');
```



```
// This will show us the array of elements
$('.a-good-old-jquery-selector').click(function(){});
```

Events

- When using javascript you have the option to use many kind of events



UI EVENTS — Occur when a user interacts with the browser's user interface (UI) rather than the web page

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

KEYBOARD EVENTS — Occur when a user interacts with the keyboard (see also input event)

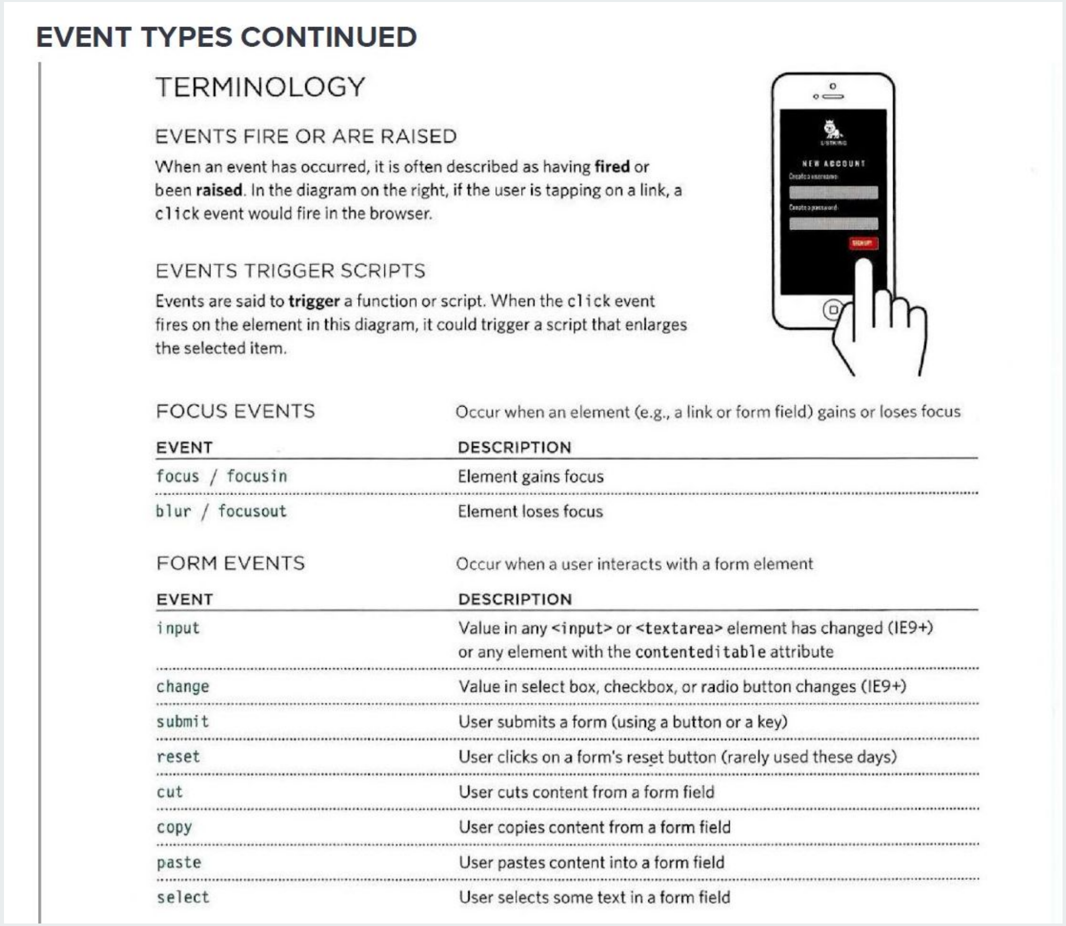
EVENT	DESCRIPTION
keydown	User first presses a key (repeats while key is depressed)
keyup	User releases a key
keypress	Character is being inserted (repeats while key is depressed)

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 mouse off an element (not on a touchscreen)

Events

- When using javascript you have the option to use many kind of events



EVENT TYPES CONTINUED

TERMINOLOGY

EVENTS FIRE OR ARE RAISED

When an event has occurred, it is often described as having **fired** or been **raised**. In the diagram on the right, if the user is tapping on a link, a `click` event would fire in the browser.

EVENTS TRIGGER SCRIPTS

Events are said to **trigger** a function or script. When the `click` event fires on the element in this diagram, it could trigger a script that enlarges the selected item.

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

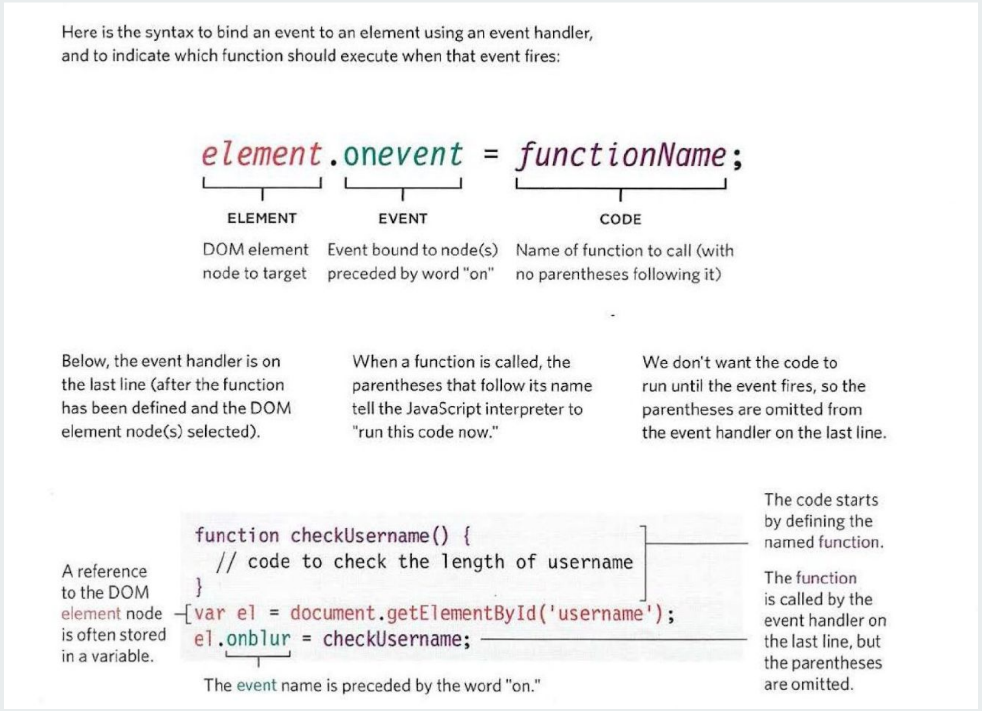
FORM EVENTS

Occur when a user interacts with a form element

EVENT	DESCRIPTION
`input`	Value in any `<input>` or `<textarea>` element has changed (IE9+) or any element with the `contenteditable` attribute
`change`	Value in select box, checkbox, or radio button changes (IE9+)
`submit`	User submits a form (using a button or a key)
`reset`	User clicks on a form's reset button (rarely used these days)
`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

Events

- This is an event handler. It is easy to set up but can only add one event to a given element



Here is the syntax to bind an event to an element using an event handler, and to indicate which function should execute when that event fires:

```
element.onevent = functionName;
```

ELEMENT
DOM element node to target
EVENT
Event bound to node(s) preceded by word "on"
CODE
Name of function to call (with no parentheses following it)

Below, the event handler is on the last line (after the function has been defined and the DOM element node(s) selected).

When a function is called, the parentheses that follow its name tell the JavaScript interpreter to "run this code now."

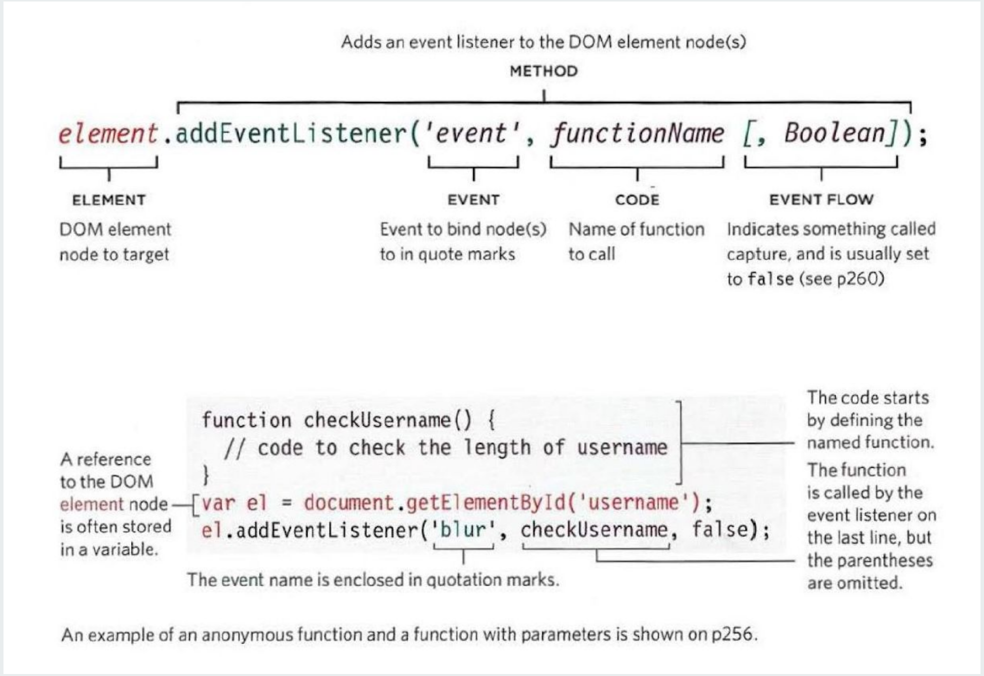
We don't want the code to run until the event fires, so the parentheses are omitted from the event handler on the last line.

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

The code starts by defining the named function.
The function is called by the event handler on the last line, but the parentheses are omitted.
A reference to the DOM element node is often stored in a variable.
The event name is preceded by the word "on."

Events

- This one is the standard event listener. It allows for multiple events to be added to the same element.


Adds an event listener to the DOM element node(s)
METHOD
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 (see p260)
A reference to the DOM element node is often stored in a variable.
The code starts by defining the named function.
The function is called by the event listener on the last line, but the parentheses are omitted.
The event name is enclosed in quotation marks.

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

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

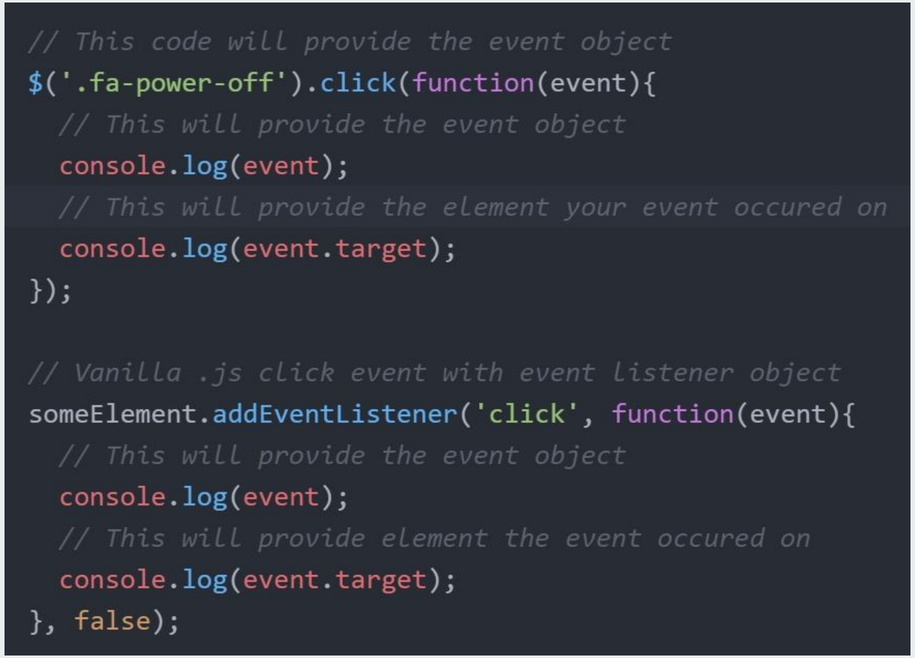
An example of an anonymous function and a function with parameters is shown on p256.

Event Object

The event object contains data that can be used in coding. Data that includes:

- location of event
- time of event
- what element was clicked or used in the event

The code on the right shows how to use/reference an event object.



```
// This code will provide the event object
$('.fa-power-off').click(function(event){
  // This will provide the event object
  console.log(event);
  // This will provide the element your event occured on
  console.log(event.target);
});

// Vanilla .js click event with event listener object
someElement.addEventListener('click', function(event){
  // This will provide the event object
  console.log(event);
  // This will provide element the event occured on
  console.log(event.target);
}, false);
```

References:

https://www.w3schools.com/jsref/met_document_queryselector.asp

https://www.w3schools.com/jsref/met_document_queryselectorall.asp

https://www.w3schools.com/js/js_htmldom_eventlistener.asp

Images referenced from Duckett:

----

http://javascriptbook.com/