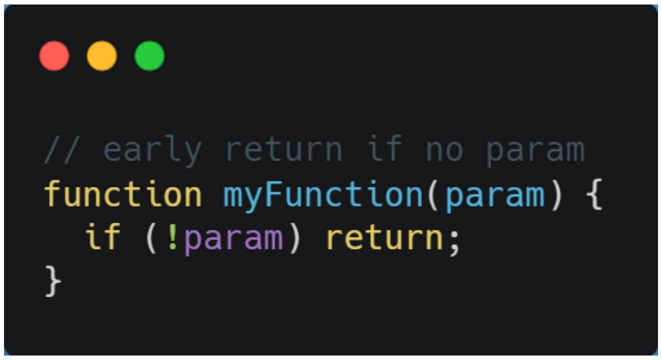
Dave’s Vanilla JavaScript Project Cheatsheet

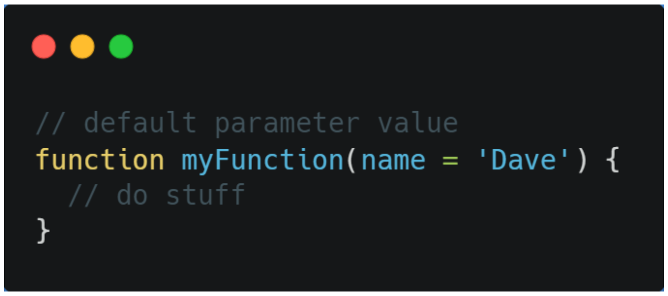
Existence of parameters:
Check if required function parameters exist and if they do not, do an early return:



```
// early return if no param
function myFunction(param) {
  if (!param) return;
}
```

Value undefined is returned.

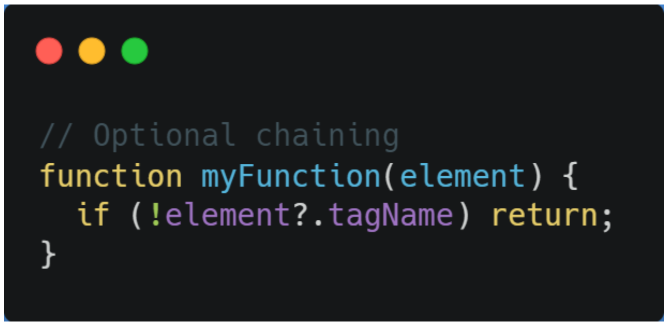
Unless a default param value is provided. Default parameter values guarantee your function will always have parameter data to work with:



```
// default parameter value
function myFunction(name = 'Dave') {
  // do stuff
}
```

If checking for existence of an object parameter AND specific object properties, use optional chaining to avoid an error. This also applies to arrays, NodeLists, and HTMLCollections when referencing the length property (array?.length) Remember, typeof array is object. HTML elements are objects, too.

Optional chaining usage when checking for an HTML element parameter:

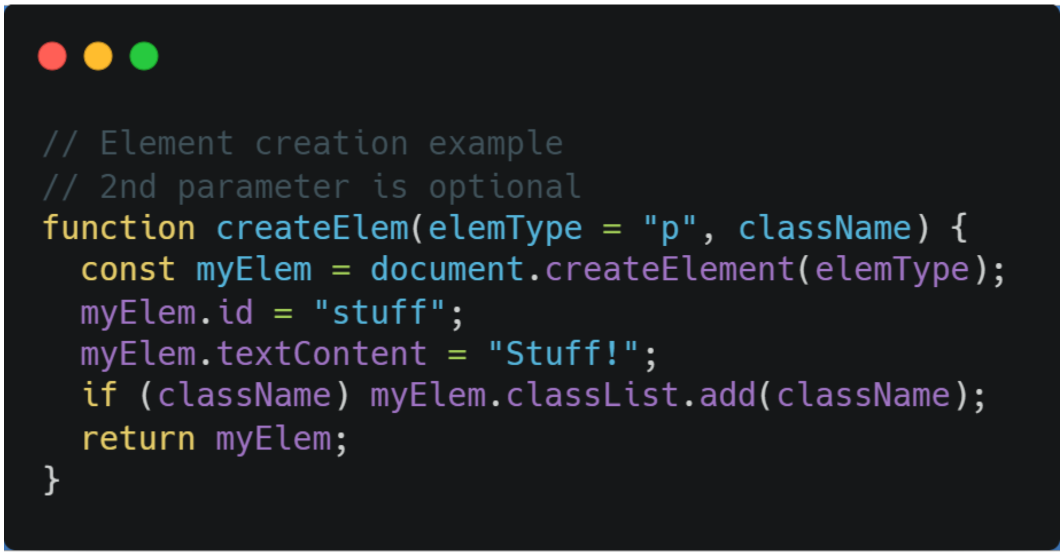


```
// Optional chaining
function myFunction(element) {
  if (!element?.tagName) return;
}
```

More examples and info on Optional Chaining in this tutorial: https://youtu.be/ULniqxZ8ueI

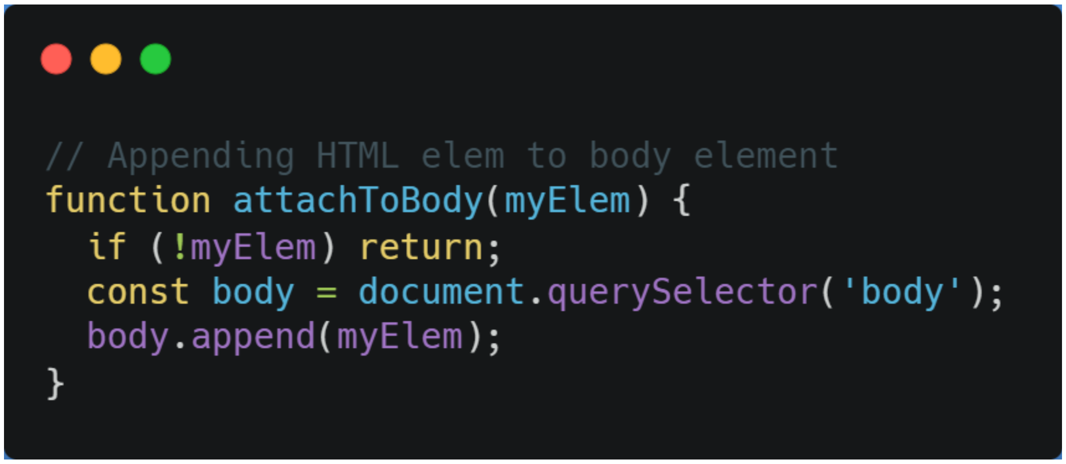
Dynamic Creation of HTML Elements:

- Use document.createElement() to create elements.
- HTML elements are objects. You can use dot notation to set their attribute values.
- Avoid use of the innerHTML property. Use the textContent property instead.
- Do not modify individual style properties. Instead apply classes defined in CSS file(s).
- You can add() and remove() classes dynamically.



```javascript
// Element creation example
// 2nd parameter is optional
function createElem(elemType = "p", className) {
  const myElem = document.createElement(elemType);
  myElem.id = "stuff";
  myElem.textContent = "Stuff!";
  if (className) myElem.classList.add(className);
  return myElem;
}
```

Appending HTML elements to the page after they are created:



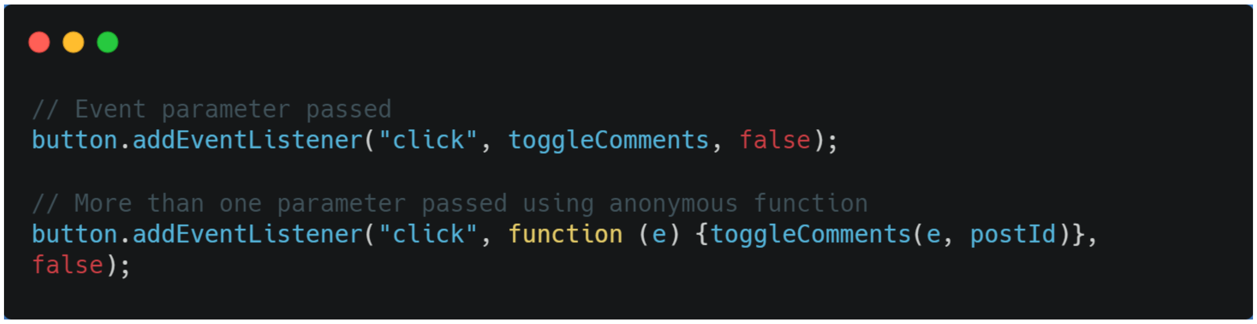
```javascript
// Appending HTML elem to body element
function attachToBody(myElem) {
  if (!myElem) return;
  const body = document.querySelector('body');
  body.append(myElem);
}
```

Append will accept more than one element: body.append(h1, p, p2)

Reference tutorial: https://youtu.be/ILcu32Nkq_I

Passing parameters with event listeners:

- There is no need to specify the event parameter if it is the only parameter. The listener will auto-magically pass the event parameter to the event handler function.
- If there is more than one parameter to pass, use an anonymous function as the event handler. It should receive the event parameter and can then pass it along with other parameters to a function called inside the anonymous function.

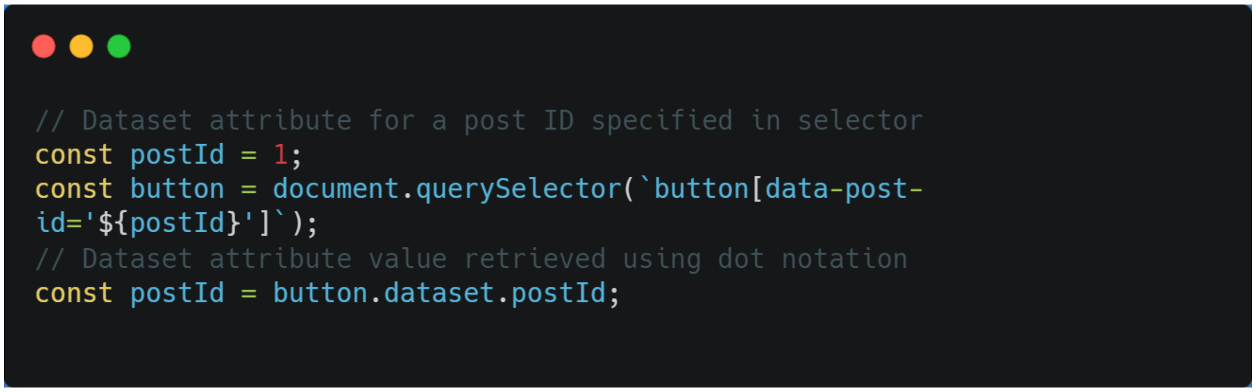


```
// Event parameter passed
button.addEventListener("click", toggleComments, false);

// More than one parameter passed using anonymous function
button.addEventListener("click", function (e) {toggleComments(e, postId)},
false);
```

Dataset attribute:

- The dataset attribute is a great place to store reference data that does not need to appear on the page… but does need to be accessed programmatically.
- The dataset attribute appears differently in the HTML document than the dot notation property that you access.

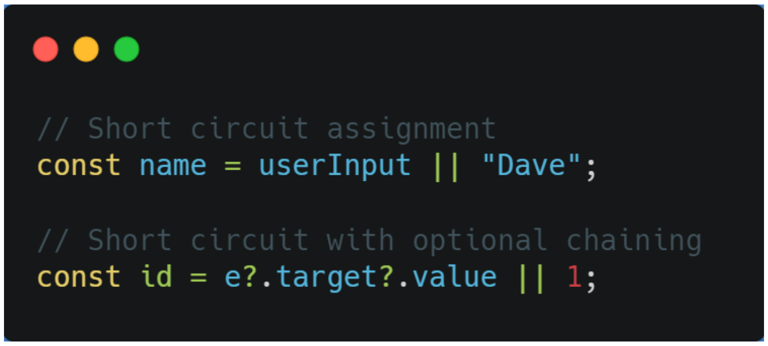


```
// Dataset attribute for a post ID specified in selector
const postId = 1;
const button = document.querySelector(`button[data-post-
id='${postId}']`);
// Dataset attribute value retrieved using dot notation
const postId = button.dataset.postId;
```

Reference: [Using data attributes - Learn web development | MDN]()

Short circuit values:

You can set a fallback value for a variable with the || operator. This means “OR”. This is also often referred to as the “pipes” operator, but most commonly known as the “short circuit”. If the first value doesn’t exist, the fallback (2nd) value is assigned to the variable.



```
// Short circuit assignment
const name = userInput || "Dave";

// Short circuit with optional chaining
const id = e?.target?.value || 1;
```