Are you starting a small-to-middle project? Going to support only *modern* browsers (i.e. IE9+)? Picking the right framework/toolkit/library/whatever for you? **You don't need to!**

Modern JS and DOM implementations offer cross-browser solutions for most use cases; there is no need for legacy compatibility code.

Let's have a look at how to write JavaScript without any library extensions:

To query a single element from the DOM tree:	`document.querySelector("#foo .bar")`
To query a list of elements from the DOM tree:	`document.querySelectorAll("p img")`
To add/remove an event listener:	`node.addEventListener("click", func)` `node.removeEventListener("click", func)`
To add/remove an event listener without losing context:	`node.addEventListener("click", this)` `node.removeEventListener("click", this)`
To stop the event bubbling:	`e.stopPropagation()`
To prevent the default event handling logic:	`e.preventDefault()`
To retrieve the element which handled the event:	`e.target`
To properly pass a callback with optional arguments:	`func.bind(this, opt1, …)`
To serialize/parse a data structure:	`JSON.stringify(obj)` `JSON.parse(str)`
To trim a string:	`" some words\n".trim()`
To manipulate CSS classes:	`node.classList.{add,remove,contains}(...)`

To send a HTTP request:	`var xhr = new XMLHttpRequest();`
	`xhr.open("get", url, true);`
	`xhr.send();`
	`xhr.onreadystatechange = cb;`

To do functional stuff:	`[].{map, forEach, filter, every, some, reduce, …}`
To retrieve the maximum available (window) size:	`[window.innerWidth, window.innerHeight]`
To animate stuff:	Use CSS transitions/animations instead!
To do vector graphics:	Just use `<svg>` elements and the correspondng JS API
To do raster graphics:	`<canvas>` and its JS API