Lesson Tuesday

Introduction to Programming (/introduction-to-programming) / JavaScript and Web Browsers (/introduction-to-programming/javascript-and-web-browsers) / JavaScript's Global Object

Text Cheat sheet

JavaScript's Global Object

- JavaScript's global scope is represented by a global object that changes based on its execution context.
- A global object is just an object that is always available and accessible.
- The execution context of our JS code is where we're running the code (for example, in a web browser or in a server).
- For JavaScript run in the browser, the global object is set to the `window` object.
- All globally scoped built-in or custom JS functions are added to the global object. In our projects, that means these functions are added to the `window` object.
    - However, only globally scoped variables declared with `var` are added to the global object.
    - We also need to be careful not to accidentally override `window` properties.

- Because the `window` object is the global object for our JS, we don't have to explicitly reference the `window` object to access its properties. For example, `window.document` can be written just as `document`, and this is the conventional way of doing things.

Convention

Going forward in LHTP lessons, we won't always explicitly include `window` when we want to access `window` properties, however, anytime it's important to know how a new object (like `document` or a global built-in JavaScript function) is connected to the `window` object we'll make that clear. Shifting to this new convention will take a bit of time to get used to, but we'll have lots of practice and you'll be given reminders.

If you prefer to type out `window.alert()` instead of `alert()` to call on this method, that is completely acceptable. The same goes for every other example we've covered in this lesson — do what's most comfortable for you and best for your learning. **In any independent project, it is your choice whether or not to omit `window` when you are accessing a property of it.**

Examples

JavaScript Global Functions

In fact, all of JavaScript's global built-in functions **or** custom functions we write (like `add()`) are properties of JavaScript's global object. Let's look at an example. Do you remember JavaScript's global built-in functions for parsing numbers? Theses ones:

```
> const myNumber = parseInt("3");
> myNumber;
3
> const myPi = parseFloat("3.14");
> myPi;
3.14
```

These global built-in JavaScript functions can be re-written and called as `window` object methods, and the functionality is the exact same:

```
> const myNumber = window.parseInt("3");
> myNumber;
3
> const myPi = window.parseFloat("3.14");
> myPi;
3.14
```

Note — there's no reason to use `window.parseInt()` instead of `parseInt()` in your projects. The point of these examples is to show you how JavaScript's global built-in functions (like `parseInt()`) and the functions we create at the global scope (like `add()`) are added to JavaScript's global object, which in the case of JavaScript run in the browser is set to the `window` object.

`window` properties

All of these `window` properties that we've learned about:

```
window.innerHeight;
window.innerWidth;
window.open();
window.location.reload();
window.location.host;
window.location.href;
window.alert();
window.prompt();
window.confirm();
window.document;
window.document.body;
window.document.head;
window.document.getElementById();
window.document.querySelectory();
```

Can all be written without first accessing the `window` object, like this:

```
innerHeight;
innerWidth;
open();
location.reload();
location.host;
location.href;
alert();
prompt();
confirm();
document;
document.body;
document.head;
document.getElementById();
document.querySelectory();
```

Previous (/introduction-to-programming/javascript-and-web-browsers/practice-accessing-the-dom)
Next (/introduction-to-programming/javascript-and-web-browsers/accessing-html-element-attributes-and-properties-in-the-

dom)

Lesson 46 of 75
Last updated more than 3 months ago.

disable dark mode

Epicodus (http://www.epicodus.com)

© 2023 Epicodus (http://www.epicodus.com/), Inc.