

Functions

Level 2 – Section 1

Issues With Flexible Function Arguments

Unexpected arguments might cause **errors** during function execution.

```
loadProfiles(["Sam", "Tyler", "Brook"]);
```



```
loadProfiles();
```

```
loadProfiles(undefined);
```

> TypeError: Cannot read property 'length' of undefined



Breaks when called with no arguments

```
function loadProfiles(userNames){  
  let namesLength = userNames.length;  
  //...  
}
```



Cannot assume userNames
will always be assigned

Manual Argument Checks Don't Scale Well

A common practice is to **check for presence** of arguments as the very first thing in the function.

```
function loadProfiles(userNames){  
  let names = typeof userNames !== 'undefined' ? userNames : [];  
  let namesLength = names.length;  
  // ...  
}
```



Too verbose and doesn't scale
well for multiple arguments

Using Default Parameter Values

Default parameter values help move **default values** from the function body to the **function signature**.

```
function loadProfiles(userNames = []){  
  let namesLength = userNames.length;  
  console.log(namesLength);  
}
```

Uses empty array as default value
when no argument is passed

Does not break when invoked with no arguments

```
loadProfiles();
```

> 0

Nor with explicit undefined as argument

```
loadProfiles(undefined);
```

> 0

The Options Object

The **options object** is a widely used pattern that allows user-defined settings to be passed to a function in the form of properties on an object.

```
setPageThread("New Version out Soon!", {  
  popular: true,  
  expires: 10000,  
  activeClass: "is-page-thread"  
});
```

Options object with 3 properties

```
function setPageThread(name, options = {}){  
  
  let popular = options.popular;  
  let expires = options.expires;  
  let activeClass = options.activeClass;  
  //...  
}
```

Options object is second argument

Assign from properties
to local variables

Issues With the Options Object

The **options object** makes it hard to know what options a function accepts.

```
setPageThread("New Version out Soon!", {  
  popular: true,  
  expires: 10000,  
  activeClass: "is-page-thread"  
});
```

```
function setPageThread(name, options = {}){  
  let popular = options.popular;  
  let expires = options.expires;  
  let activeClass = options.activeClass;  
  //...  
}
```



Unclear what options this function expects just by looking at its signature

Boilerplate code

Using Named Parameters

Using **named parameters** for optional settings makes it easier to understand how a function should be invoked.



```
function setPageThread(name, { popular, expires, activeClass }){  
  console.log("Name: ", name);  
  console.log("Popular: ", popular);  
  console.log("Expires: ", expires);  
  console.log("Active: ", activeClass);  
}
```

Now we know which arguments are available

Local variables

```
setPageThread("New Version out Soon!", {  
  popular: true,  
  expires: 10000,  
  activeClass: "is-page-thread"  
});
```

```
> Name: New Version out Soon!  
> Popular: true  
> Expires: 10000  
> Active: is-page-thread
```



Omitting Certain Arguments on Call

It's okay to omit **some options** when invoking a function with named parameters.

```
function setPageThread(name, { popular, expires, activeClass }){  
  console.log("Name: ", name);  
  console.log("Popular: ", popular);  
  console.log("Expires: ", expires);  
  console.log("Active: ", activeClass);  
}
```

popular is the only named argument being passed

```
setPageThread("New Version out Soon!", {  
  popular: true  
});
```

```
> Name: New Version out Soon!  
> Popular: true  
> Expires: undefined  
> Active: undefined
```

No value assigned to
remaining parameters



Don't Omit All Named Arguments on Call



It's **NOT okay** to omit the options argument altogether when invoking a function with named parameters when no default value is set for them.



```
function setPageThread(name, { popular, expires, activeClass }){  
  
  console.log("Name: ", name);  
  console.log("Popular: ", popular);  
  console.log("Expires: ", expires);  
  console.log("Active: ", activeClass);  
}
```

```
setPageThread("New Version out Soon!");
```


Invoking this function without its second argument breaks our code



```
> TypeError: Cannot read property  
  'popular' of undefined
```

Setting a Default for Named Parameters


Setting a **default value** for the entire options argument allows this parameter to be omitted during function calls.



```
function setPageThread(name, { popular, expires, activeClass } = {}){  
  console.log("Name: ", name);  
  console.log("Popular: ", popular);  
  console.log("Expires: ", expires);  
  console.log("Active: ", activeClass);  
}
```

```
setPageThread("New Version out Soon!");
```

We can now safely invoke this function without its second argument



```
> Name: New Version out Soon!  
> Popular: undefined  
> Expires: undefined  
> Active: undefined
```