WDD330

Ribeka Nanjo

W02 Reading (L4)

Testing and Debugging

1. Debugging in the Browser

The trusty Alert
- alert(): it shows a dialog at certain points
e.g f
unction amIOldEnough(age){
if (age = 12) {
alert(age);
return 'No, sorry.';
} else if (age < 18) {
return 'Only if you are accompanied by an adult.';
} else {
return 'Yep, come on in!';
}}

- console(): can be used to log the value of variables
e.g
function amIOldEnough(age){
console.log(age);
if (age < 12) {
console.log(`In the if with ${age}`);
return 'No, sorry.';
} else if (age < 18) {
console.log(`In the else-if with ${age}`);
return 'Only if you are accompanied by an adult.';
} else {
console.log(`In the else with ${age}`);
return 'Yep, come on in!';
}}

2. Debugging Tools

- Firefox
- Edge
- Chrome
- Safari

debugger- break point that will pause the execution of the code. We can also hover over any variables to see what value they hold at that point. To start it, click the "play" button.

3. Error Object

4. Throwing Expectations

throw- throw our own expectations using the statement. This will allow for any problems in our codes to be highlighted and dealt with.

5. Exception Handling

try- we can wrap it in a try block if we suspect a piece of code will result in an exception
e.g

```
function imaginarySquareRoot(number) {
'use strict';
try {
return String(squareRoot(number));
} catch(error) {
return squareRoot(-number)+'i';
}
}
```

If anexception occurs it will pass the error object that is thrown onto a catch block.

```
function imaginarySquareRoot(number) {
'use strict';
let answer;
try {
answer = String(squareRoot(number));
} catch(error) {
```

```
answer = squareRoot(-number)+"i";
} finally {
return `+ or - ${answer}`;
}}
```

6. Test

1. Write tests (that initially fail)
2. Write code to pass the tests
3. Refactor the code
4. Test refactored code
5. Write more tests for new features